# MULTIMODAL AND EMBODIED LEARNING WITH LANGUAGE AS THE ANCHOR

Hyounghun Kim

A thesis submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2022

Approved by:

Mohit Bansal

Trung Bui

Henry Fuchs

Jasleen Kaur

David Seunghyun Yoon

# ABSTRACT

Hyounghun Kim: Multimodal and Embodied Learning with Language
as the Anchor
(Under the direction of Mohit Bansal)

Since most worldly phenomena can be expressed via language, language is a crucial medium for transferring information and integrating multiple information sources. For example, humans can describe what they see, hear and feel, and also explain how they move with words. Conversely, humans can imagine scenes, sounds, and feelings, and move their body from language descriptions. Therefore, language plays an important role in solving machine learning (ML) and artificial intelligence (AI) problems with multimodal input sources. This thesis studies how different modalities can be integrated with language in multimodal learning settings as follows.

First, we explore the possibility to integrate external information from the textual description about an image into a visual question answering system which integrates the key words/phrases in paragraph captions in semi-symbolic form, to make the alignment between features easier. We expand the direction to a video question answering task. We employ dense captions, which generate object-level descriptions of an image, to help localize the key frames in a video clip for answering a question.

Next, we build benchmarks to evaluate embodied agents to perform tasks according to natural language instruction from humans. We introduce a new instruction-following navigation and object assembly system, called ArraMon in which agents follow the natural language instructions to collect an object and put it in a target location, requiring agents to deeply understand referring expressions and the concept of direction from the egocentric perspective. We also suggest a new task setup for the useful Cooperative Vision-and-Dialog Navigation (CVDN) dataset. We analyze scoring behaviors of models and find issues from the existing Navigation from Dialog History

(NDH) task and propose a more realistic and challenging task setup, called NDH-Full which better appreciates the purpose of the CVDN dataset.

Finally, we explore AI assistant systems which help humans with different tasks. We introduce a new correctional captioning dataset on human body pose, called FixMyPose, to encourage the ML/AI community to build such guidance systems that require models to learn to distinguish different levels of pose difference to describe desirable pose change. Also, we introduce a new conversational image search and editing assistant system, called CAISE, in which an agent helps a user to search images and edit them by holding a conversation.

To my family.

## ACKNOWLEDGEMENTS

First, I would like to thank my advisor Prof. Mohit Bansal and appreciate his effort to support my journey of research as a Ph.D. student. His advising helped me grow as a researcher intrinsically and extrinsically, which will be the ground foundation of my future endeavor.

I would like to thank Dr. Trung Bui, Prof. Henry Fuchs, Prof. Jasleen Kaur, and Dr. David Seunghyun Yoon for being my committee members and providing constructive and valuable feedback on my thesis. I also thank my co-workers for giving great opportunities to work with them on various topics. Especially, I appreciate Abhay Zala for his effort for collaborations on multiple papers. Thank you UNC MURGe-Lab members. I am very proud of being a member of this great team.

Finally, I want to say thank you to my family, especially my wife Sewon and my son Ian. They have been enduring a long challenging time together by me being my support for everything.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| RL | Reinforcement Learning |
| VQA | Visual Question Answering |

# CHAPTER 1: INTRODUCTION

Since most worldly phenomena can be expressed via language, language is a crucial medium for transferring information and integrating multiple information sources. For example, humans can describe what they see, hear and feel, and also explain how they move with words. Conversely, humans can imagine scenes, sounds, and feelings, and move their body from language descriptions. Therefore, language plays an important role in solving machine learning (ML) and artificial intelligence (AI) problems with multimodal input sources. This thesis studies how different modalities can be integrated with language in multimodal learning settings. To be specific, we focus on learning computer vision (visual and video question answering), embodied agents/robotics (vision-and-language navigation), and multimodal AI assistant systems (body pose correction feedback and image editing assistants) tasks with language.

First, we explore the possibility to integrate external information from the textual description of an image into a visual question answering system. Representations from similar types of information would be closer to one another than other types. Therefore, converting a type of data to a target data type could help facilitate better alignment with the data of the same type. In our work (Kim and Bansal, 2019), we apply the approach to visual question answering (VQA) task (Antol et al., 2015; Krishna et al., 2017). In VQA task, clues from images for answering questions are provided in visual data type while the questions are given in a textual format, thus, models can benefit from being provided with extra textual descriptions of the images. To implement, we employ paragraph captions (Krause et al., 2017), which contain diverse aspects of an image, as the external information source, and proposed the Visual and Textual Question Answering (VTQA) model, which integrates the key words/phrases in paragraph captions in semi-symbolic form, to make the alignment between features easier through three different levels of fusions. Results from the extensive experiments and analysis show the proposed approach help extract useful clues and improve the model's performance. We expand the direction (semi-symbolic matching) to a video question answering task (Kim et al., 2020a). Because a video clip consists of multiples frames, localizing frames with crucial clues is important in video question answering tasks. We employ dense captions (Johnson et al., 2016), which generate detailed object-level descriptions of an image, to help localize the key frames in a video clip for answering a question. The proposed model is further equipped with the dual-level attention and the gating mechanism, and trained via newly proposed losses for improving performance. The proposed model outperforms the state-of-the-art models and also shows the more balanced performance across all TV shows.

Using natural language as an interface for communicating with embodied agents or robots is getting important. Therefore, developing embodied agents/robots that understand human language is also obtaining attention. Accordingly, we introduce a new instruction-following navigation and object assembly system, called ARRAMON (Kim et al., 2020b). In ARRAMON, agents follow the natural language instructions, which have rich linguistic properties, to collect an object and put it in a target location which is also designated by language instructions. Therefore, agents are required to deeply understand referring expressions and the concept of direction from the egocentric perspective to successfully perform this challenging task. We also suggest a new task setup for the useful Cooperative Vision-and-Dialog Navigation (CVDN) dataset (Thomason et al., 2019) along with a strong baseline model (Kim et al., 2021a). We analyze scoring behaviors of models and found issues (i.e., insufficient language supervision and a mismatched evaluation metric) from the existing Navigation from Dialog History (NHD) task (Thomason et al., 2019) and propose a more realistic and challenging task setup, called NDH-FULL. In NDH-FULL, an agent is given a full-length dialogue and asked to navigate to reach the goal region by referring to proper guidance from the human oracle in the dialogue, which better appreciates the purpose of the CVDN dataset. Experiments and analysis show that NDH-FULL provides full language supervision towards the goal regions, but it is still challenging to finish the task due to long dialogues and trajectories.

These days, AI assistant systems are already deployed in our daily lives. They help humans in various tasks communicating via natural language. On the other hand, as demands for participating in more diverse activities are increasing, AI assistant systems are asked to obtain more abilities to help humans with challenging tasks. Thus, we explore the potential directions that the AI assistant systems should pursue. Recently, interest in remote physical learning is getting increased and automated feedback systems are required accordingly. Therefore, we introduce a new correctional captioning dataset on human body pose, called FIXMYPOSE, to encourage the ML/AI community to build such guidance systems (Kim et al., 2021b). From the FIXMYPOSE dataset, we propose two tasks, pose-correction-captioning and target-pose-retrieval. In the pose-correction-captioning task, the model is asked to generate verbal guidance given 'current' and 'target' images which show different poses of a 3D avatar. In the pose-correction-captioning, the model is asked to select a correct target image given the current image and the correctional description. Both tasks are challenging because models should learn to distinguish different levels of pose difference to perform successfully, thus can be new types of benchmarks for evaluating referring expression and spatial relation understanding. We also introduce a new image search and editing assistant system called, CAISE, in which an agent helps a user to search images and edit them by holding a conversation (Kim et al., 2022). In CAISE, given dialogue history and a user request, the model is asked to generate an executable search/editing command. We propose the generator-extractor baseline model for this task, which can adaptively select the source of the next token (i.e., from the vocabulary or from textual/visual contexts) for the executable command. The proposed system suggests a new direction for automated image editing assistant systems encouraging the development of more complex real-world applications for non-expert users.

## 1.1 Thesis Statement

Language is the anchor for integrating diverse information and communication with AI systems through its flexibility, and it plays a crucial role in building effective multimodal and embodied machine learning and general intelligent agents.

## 1.2 Overview of Chapters

The remainder of this dissertation is organized into eight chapters. Chapter 2 discusses the related work around visual/video question answering, image captioning, instruction-following navigation and object manipulation, human pose, and automated image editing systems. Chapter 3 and 4 present our work on the semi-symbolic matching approach for visual and video question answering. Chapter 5 and 6 present our new instruction-following dataset/task and a new task setup we proposed from CVDN dataset. Chapter 7 and 8 present our new AI assistant systems on human body pose correctional guidance, and image search and editing. Finally, Chapter 9 summarizes the contributions and discusses the future work.

# CHAPTER 2: BACKGROUND AND RELATED WORK

In this chapter, we present the background and related work of visual/video question answering, image captioning, instruction-following navigation and object manipulation, human pose, and automated image editing systems.

## 2.1 Visual/Video Question Answering and Image Captioning

Understanding visual information conditioned on language is an important ability for an agent who is supposed to have integrated intelligence. Many tasks have been proposed to evaluate such ability, and visual question answering is one of those tasks (Antol et al., 2015; Lu et al., 2016; Fukui et al., 2016; Xu and Saenko, 2016; Yang et al., 2016a; Zhu et al., 2016; Goyal et al., 2017; Anderson et al., 2018a). To perform this task successfully, models should understand the meaning of the given question and extract relevant clues from images, and align it with the question feature. Recently, beyond question answering on a single image, attention to understanding and extracting information from a sequence of images, i.e., a video, is rising (Tapaswi et al., 2016; Maharaj et al., 2017; Kim et al., 2017; Jang et al., 2017; Lei et al., 2018; Zadeh et al., 2019; Lei et al., 2020b; Garcia et al., 2020). Answering questions on videos requires an understanding of temporal information as well as spatial information, making video question answering tasks more challenging than single-image question answering tasks.

Another thread of research which deals with combined visual and language problem is the translation of visual contents to natural language. Describing image contents in natural language has been actively studied (Xu et al., 2015; Yang et al., 2016b; Rennie et al., 2017; Lu et al., 2017; Anderson et al., 2018a; Melas-Kyriazi et al., 2018; Yao et al., 2018). This progress has been encouraged by the introduction of large-scale captioning datasets (Hodosh et al., 2013; Lin et al., 2014; Plummer et al., 2015; Krishna et al., 2017). While usual single-sentence captions only describe the main content or topic of an image, dense and paragraph captions focus on multiple regions to give diverse aspects of the image (Johnson et al., 2016; Krause et al., 2017). Recently, more diverse image captioning tasks, which consider two images and describe the difference between them, have been introduced (Jhamtani and Berg-Kirkpatrick, 2018; Tan et al., 2019a; Park et al., 2019; Forbes et al., 2019). For this image difference captioning task, models should identify changes in objects and regions and express how they are altered using their spatial relations, attributes, etc.

In this dissertation, we show that textual descriptions from image captions help better understand visual scenes via intermediate-symbolic matching for improving image/video question answering systems.

## 2.2 Vision-and-Language Navigation & Object Manipulation Task.

Recently, Vision-and-Language Navigation (VLN) tasks, in which agents follow natural language instructions to navigate through an environment, have been actively studied in research communities (MacMahon et al., 2006; Mooney, 2008; Chen and Mooney, 2011; Tellex et al., 2011; Mei et al., 2016; Hermann et al., 2017; Brahmbhatt and Hays, 2017; Mirowski et al., 2018; Anderson et al., 2018b; Misra et al., 2018; Blukis et al., 2018; Das et al., 2018; Cirik et al., 2018; de Vries et al., 2018; Blukis et al., 2019; Thomason et al., 2019; Nguyen et al., 2019; Nguyen and Daumé III, 2019; Chen et al., 2019; Jain et al., 2019; Shridhar et al., 2020; Qi et al., 2020; Hermann et al., 2020; Berg et al., 2020; Zhu et al., 2020a). To encourage the exploration of this challenging research topic, multiple simulated environments have been introduced. Synthetic (Kempka et al., 2016; Beattie et al., 2016; Kolve et al., 2017; Brodeur et al., 2017; Wu et al., 2018; Savva et al., 2017; Zhu et al., 2017; Yan et al., 2018a; Shah et al., 2018; Puig et al., 2018) as well as real-world and image-based environments (Brahmbhatt and Hays, 2017; Mirowski et al., 2018; Anderson et al., 2018b; Xia et al., 2018; Cirik et al., 2018; de Vries et al., 2018; Chen et al., 2019; Savva et al., 2019) have been used to provide agents with diverse and complement training environments. Object manipulation and configuration is another subject that has been studied along with language and vision grounding (Bisk et al., 2016; Wang et al., 2016b; Li et al., 2016; Bisk et al., 2018).

There have been only a few recent efforts to combine the traditional navigation task with other tasks. Touchdown (Chen et al., 2019) combines navigation and object referring expression resolution, REVERIE (Qi et al., 2020) performs remote referring expression comprehension, while ALFRED (Shridhar et al., 2020) combines indoor navigation and household manipulation. Therefore, it is required to build more challenging and integrated training environments to train and evaluate agents in more diverse and realistic scenarios. In this dissertation, we introduce a new complementary task, called ARRAMON that merges navigation in a complex outdoor space with object referring expression comprehension and assembling tasks that require spatial relation understanding in an interweaved temporal style, in which the two tasks alternate for multiple turns leading to cascading error effects.

## 2.3 Human Pose

Human pose estimation and action recognition have been a long-standing topic in the research community (Johnson and Everingham, 2010, 2011; Andriluka et al., 2014; Toshev and Szegedy, 2014; Wei et al., 2016; Andriluka et al., 2018; Yan et al., 2018b; Zhao et al., 2019; Cao et al., 2019; Sun et al., 2019; Verma et al., 2020; Rong et al., 2020). Recently, researchers are also focusing on generation tasks which generate a body pose sequence from an input of a different type from another modality such as audio or spoken language (Shlizerman et al., 2018; Tang et al., 2018; Lee et al., 2019; Zhuang et al., 2020; Saunders et al., 2020). However, there have been no research attempts on text generation based on pose correction. In this dissertation, we introduce a novel dataset, called FIXMYPOSE which consists of image pairs (a "current" image which contains a wrong pose and a "target" image which contains a correct target pose) and corresponding correctional descriptions that explain how to correct the wrong pose to the desired pose. The proposed dataset will encourage the community to explore this new pose correctional guidance topic.

### 2.4 Automated Image Editing Systems

There have been some prior efforts to automate image editing programs. The research on image editing has been focused on intent identification (Manuvinakurike et al., 2018c), request to actionable command mapping (Manuvinakurike et al., 2018b; Lin et al., 2018), dialogue act labeling (Manuvinakurike et al., 2018a), low-level image edit requests (Lin et al., 2020), description to editing (Shi et al., 2020), or editing to description (Tan et al., 2019a). Also, language-based image editing (Shinagawa et al., 2017; Chen et al., 2018; El-Nouby et al., 2019; Fu et al., 2020) focuses on an image generation task setup. However, there have been relatively few studies that pursue end-to-end conversational image editing agent systems. In this dissertation, we introduce a new dataset, called CAISE in which a user and an assistant hold a conversation in natural language about image search and editing. It supports the direct deployment of conversational image editing assistant systems by incorporating executable commands, and also integrates image search functionality so as to make it more comprehensively useful.

# CHAPTER 3: SEMI-SYMBOLIC MATCHING FOR VISUAL QUESTION ANSWERING

## 3.1 Introduction

Understanding visual information along with natural language has been studied in different ways. In visual question answering (VQA) (Antol et al., 2015; Goyal et al., 2017; Lu et al., 2016; Fukui et al., 2016; Xu and Saenko, 2016; Yang et al., 2016a; Zhu et al., 2016; Anderson et al., 2018a), models are trained to choose the correct answer given a question about an image. On the other hand, in image captioning tasks (Karpathy and Fei-Fei, 2015; Johnson et al., 2016; Anderson et al., 2018a; Krause et al., 2017; Liang et al., 2017; Melas-Kyriazi et al., 2018), the goal is to generate sentences which should describe a given image. Similar to the VQA task, image captioning models should also learn the relationship between partial areas in an image and the generated words or phrases. While these two tasks seem to have different directions, they have the same purpose: understanding visual information with language. If their goal is similar, can the tasks help each other?

In this work, we propose an approach to improve a VQA model by exploiting textual information from a paragraph captioning model. Suppose you are assembling furniture by looking at a visual manual. If you are stuck at a certain step and you are given a textual manual which more explicitly describes the names and shapes of the related parts, you could complete that step by reading this additional material and also by comparing it to the visual counterpart. With a similar intuition, paragraph-style descriptive captions can more explicitly (via intermediate symbolic representations) explain what objects are in the image and their relationships, and hence VQA questions can be answered more easily by matching the textual information with the questions.

We provide a VQA model with such additional 'textual manual' information to enhance its ability to answer questions. We use descriptive captions generated from a paragraph captioning

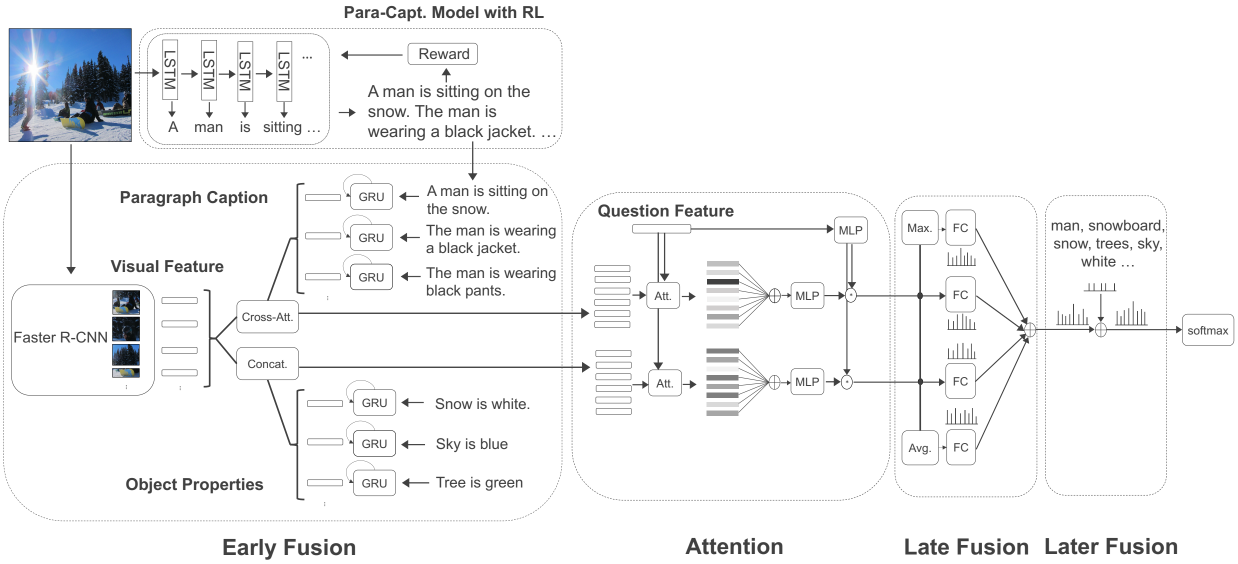


Figure 3.1: VTQA Architecture: Early, Late, and Later Fusion between the Vision and Paragraph Features.

model which capture more detailed aspects of an image than a single-sentence caption (which only conveys the most obvious or salient single piece of information). We also extract properties of objects, i.e., names and attributes from images to create simple sentences in the form of “[object name] is [attribute]”. Our VTQA model takes these paragraph captions and attribute sentences as input in addition to the standard input image features. The VTQA model combines the information from text and image with early fusion, late fusion, and later fusion. With early fusion, visual and textual features are combined via cross-attention to extract related information. Late fusion collects the scores of candidate answers from each module to come to an agreement. In later fusion, expected answers are given an extra score if they are in the recommendation list which is created with properties of detected objects. Empirically, each fusion technique provides complementary gains from paragraph caption information to improve VQA model performance, overall achieving significant improvements over a strong baseline VQA model. We also present several ablation studies and attention visualizations.

## 3.2 Models

The basic idea of our approach is to provide the VQA model with extra text information from paragraph captions and object properties (see Fig. 3.1).

### 3.2.1 Paragraph Captioning Model

Our paragraph captioning module is based on Melas-Kyriazi et al. (2018)'s work, which uses CIDEr (Vedantam et al., 2015) directly as a reward to train their model. They make the approach possible by employing self-critical sequence training (SCST) (Rennie et al., 2017). However, only employing RL training causes repeated sentences. As a solution, they apply $n$-gram repetition penalty to prevent the model from generating such duplicated sentences. We adopt their model and approach to generate paragraph captions.

### 3.2.2 VTQA Model

#### 3.2.2.1 Features

**Visual Features**: We adopt the bottom-up and top-down VQA model from Anderson et al. (2018a), which uses visual features from the salient areas in an image (bottom-up) and gives them weights using attention mechanism (top-down) with features from question encoding. Following Anderson et al. (2018a), we also use Faster R-CNN (Ren et al., 2015) to get visual features $V \in \mathbb{R}^{O \times d}$, where $O$ is #objects detected and $d$ is the dimension of each visual feature of the objects.

**Paragraph Captions**: These provide diverse aspects of an image by describing the whole scene. We use GloVe (Pennington et al., 2014) for the word embeddings. The embedded words are sequentially fed into the encoder, for which we use GRU (Cho et al., 2014), to create a sentence representation, $s_i \in \mathbb{R}^d$: $s_i = \text{ENC}_{sent}(w_{0:T})$, where $T$ is the number of words. The paragraph feature is a matrix which contains each sentence representation in each row, $P \in \mathbb{R}^{K \times d}$, where $K$ is the number of sentences in a paragraph.

**Object Property Sentences**: The other text we use is from properties of detected objects in images (name and attribute), which can provide explicit information about the corresponding object to a VQA model. We create simple sentences like, "[object name] is [attributes]". We then obtain sentence representations by following the same process as what we do with the paragraph captions above. Each sentence vector is then attached to the corresponding visual feature, like 'name tag', to allow the model to identify objects in the image and their corresponding traits.

#### 3.2.2.2 Three Fusion Levels

**Early Fusion**: In the early fusion stage, visual features are fused with paragraph caption and object property features to extract relevant information. For visual and paragraph caption features, cross-attention is applied to get similarity between each component of visual features (objects) and a paragraph caption (sentences). We follow Seo et al. (2017)'s approach to compute the similarity matrix, $S \in \mathbb{R}^{O \times K}$. From the similarity matrix $V^p = \text{softmax}(S^T)V$ and the new paragraph representation, $P^f$ is obtained by concatenating $P$ and $P * V^p$: $P^f = [P; P * V^p]$, where * is element-wise product operation. For visual feature and object property feature $C$, they are already aligned and the new visual feature $V^f$ becomes $V^f = [V; V * C]$. Given the fused representations, the attention mechanism is applied over each row of the representations to weight more relevant features to the question.

$$a_i = w_a^T(\text{ReLU}(W_{sa}s_i^f) * \text{ReLU}(W_{qa}q)) \tag{3.1}$$

$$\alpha = \text{softmax}(a) \tag{3.2}$$

where, $s_i^f$ is a row vector of new fused paragraph representation and $q$ is the representation vector of a question which is encoded with GRU unit. $w_a^T$, $W_{sa}$, and $W_{qa}$ are trainable weights. Given the attention weights, the weighted sum of each row vector, $s_i^f$ leads to a final paragraph vector $p = \sum_{i=1}^{K} \alpha_i s_i^f$. The paragraph vector is fed to a nonlinear layer and combined with question vector by element-wise product.

$$p^q = \text{ReLU}(W_p p) * \text{ReLU}(W_q q) \tag{3.3}$$

$$L_p = \text{classifier}(p^q) \tag{3.4}$$

where $W_p$ and $W_q$ are trainable weights, and $L_p$ contains the scores for each candidate answer. The same process is applied to the visual features to obtain $L_v = \text{classifier}(v^q)$.

**Late Fusion**: In late fusion, logits from each module are integrated into one vector. We adopt the approach of Wang et al. (2016a). Instead of just adding the logits, we create two more vectors by max pooling and averaging those logits and add them to create a new logit $L_{new} = L_1 + L_2 + ... + L_n + ... + L_{max} + L_{avg}$, where $L_n$ is $n$th logit, and $L_{max}$ and $L_{avg}$ are from max-pooling and averaging all other logits. The intuition of creating these logits is that they can play as extra voters so that the model can be more robust and powerful.

**Answer Recommendation or 'Later Fusion'**: Salient regions of an image can draw people's attention and thus questions and answers are much more likely to be related to those areas. Objects often denote the most prominent locations of these salient areas. From this intuition, we introduce a way to directly connect the salient spots with candidate answers. We collect properties (name and attributes) of all detected objects and search over answers to figure out which answer can be extracted from the properties. Answers in this list of expected answers are given extra credit to enhance the chance to be selected. If logit $L_{\text{before}}$ from the final layer contains scores of each answer, we want to raise the scores to logit $L_{\text{after}}$ if the corresponding answers are in the list $l_c$:

$$\begin{aligned} L_{\text{before}} &= \{a_1, a_2, ..., a_n, ..\} \\ L_{\text{after}} &= \{\hat{a}_1, \hat{a}_2, ..., \hat{a}_n, ..\} \end{aligned} \tag{3.5}$$

$$\hat{a}_n = \begin{cases} a_n + c \cdot \text{std}(L_{before}) & \text{if } n \in l_c \\ a_n & \text{otherwise} \end{cases} \tag{3.6}$$

where the std$(\cdot)$ operation calculates the standard deviation of a vector and $c$ is a tunable parameter. $l_c$ is the list of the word indices of detected objects and their corresponding attributes. The indices of the objects and the attributes are converted to the indices of candidate answers.

### 3.3 Experimental Setup

**Paragraph Caption**: We use paragraph annotations of images from Visual Genome (Krishna et al., 2017) collected by Krause et al. (2017), since this dataset is the only dataset (to our knowledge) that annotates long-form paragraph image captions. We follow the dataset split of 14,575 / 2,487 / 2,489 (train / validation / test).

**Visual Question Answering Pairs**: We also use the VQA pairs dataset from Visual Genome so as to match it with the provided paragraph captions. We almost follow the same image dataset split as paragraph caption data, except that we do not include images that do not have their own question-answer pairs in the train and evaluation sets. The total number of candidate answers is 177,424. Because that number is too huge to train, we truncate the question-answer pairs whose answer's frequency are under 30, which give us a list of 3,453 answers. So, the final number of question-answering pairs are 171,648 / 29,759 / 29,490 (train / validation / test).

**Training Details**: Our hyperparameters are selected using validation set. The size of the visual feature of each object is set to 2048 and the dimension of the hidden layer of question encoder and caption encoder are 1024 and 2048 respectively. We use AdaMax (Kingma and Ba, 2015) for the optimizer and a learning rate of 0.002. We modulate the final credit, which is added to the final logit of the model, by multiplying a scalar value $c$ (we tune this to 1.0).

### 3.4 Results, Ablations, and Analysis

**VQA vs. VTQA** As shown in Table 3.1, our VTQA model increases the accuracy by 1.92% from the baseline VQA model for which we employ Anderson et al. (2018a)'s model and apply multi-modal factorized bilinear pooling (MFB) (Yu et al., 2017). This implies that our textual

|  | Model | Test accuracy (%) |
|---|---|---|
| 1 | VQA baseline | 44.68 |
| 2 | VQA + MFB baseline | 44.94 |
| 3 | VTQA (EF+LF+AR) | 46.86 |

Table 3.1: Our VTQA model significantly outperforms ($p$ <0.001) the strong baseline VQA model (we do not apply MFB to our VTQA model, since it does not work for the VTQA model).

|  | Model | Val accuracy (%) |
|---|---|---|
| 1 | VTQA + EF (base model) | 45.41 |
| 2 | VTQA + EF + LF | 46.36 |
| 3 | VTQA + EF + AR | 46.95 |
| 4 | VTQA + EF + LF + AR | 47.60 |

Table 3.2: Our early (EF), late (LF), and later fusion (or Answer Recommendation AR) modules each improves the performance of our VTQA model.

data helps improve VQA model performance by providing clues to answer questions. We run each model five times with different seeds and take the average value of them. For each of the five runs, our VTQA model performs significantly better ($p < 0.001$) than the VQA baseline model.

**Late Fusion and Later Fusion Ablations** As shown in row 2 of Table 3.2, late fusion improves the model by 0.95%, indicating that visual and textual features complement each other. As shown in row 3 and 4 of Table 3.2, giving an extra score to the expected answers increases the accuracy by 1.54% from the base model (row 1) and by 1.24% from the result of late fusion (row 2), respectively. This could imply that salient parts (in our case, objects) can give direct cues for answering questions.[1]

**Ground-Truth vs. Generated Paragraphs** We manually investigate (300 examples) how many questions can be answered only from the ground-truth (GT) versus generated paragraph (GenP) captions. We also train a TextQA model (which uses cross-attention mechanism between question and caption) to evaluate the performance of the GT and GenP captions. As shown in Table

[1] Object Properties: Appending the encoded object properties to visual features improves the accuracy by 0.15% (47.26 vs. 47.41). This implies that incorporating extra textual information into visual features could help a model better understand the visual features for performing the VQA task.

| | Model | Val accuracy (%) |
|---|---|---|
| 1 | TextQA with GT | 43.96 |
| 2 | TextQA with GenP | 42.07 |

Table 3.3: TextQA with GT model outperforms TextQA with GenP (we run each model five times with different seeds and average the scores. GT: Ground-Truth, GenP: Generated Paragraph).

| | Human Eval. | Accuracy (%) |
|---|---|---|
| 1 | with GT | 55.00 |
| 2 | with GenP | 42.67 |

Table 3.4: Human evaluation only with paragraph captions and questions of the validation dataset. Human evaluation with GT shows better performance than human evaluation with GenP.

3.3, the GT captions can answer more questions correctly than GenP captions in TextQA model evaluation. Human evaluation with GT captions also shows better performance than with GenP captions as seen in Table 3.4. However, the results from the manual investigation have around 12% gap between GT and generated captions, while the gap between the results from the TextQA model is relatively small (1.89%). This shows that paragraph captions can answer several VQA questions but our current model is not able to extract the extra information from the GT captions. This allows future work: (1) the TextQA/VTQA models should be improved to extract more information from the GT captions; (2) paragraph captioning models should also be improved to generate captions closer to the GT captions.[2]

**Attention Analysis** Finally, we also visualize the attention over each sentence of an input paragraph caption w.r.t. a question. As shown in Figure 3.2, a sentence which has a direct clue for a question get much higher weights than others. This explicit textual information helps a VQA model handle what might be hard to reason about only-visually, e.g., ‘two (2) cows’. Please see section A.1 for more attention visualization examples.

[2] We also ran our full VTQA model with the ground truth (GT) paragraph captions and got an accuracy value of 48.04% on the validation dataset (we ran the model five times with different seeds and average the scores), whereas the VTQA result from generated paragraph captions was 47.43%. This again implies that our current VTQA model is not able to extract all the information enough from GT paragraph captions for answering questions, and hence improving the model to better capture clues from GT captions is useful future work.

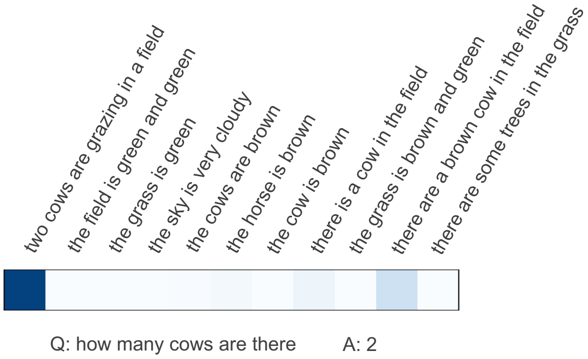


Figure 3.2: Attention Visualization for an example answered correctly by our model.

## 3.5 Conclusion

We presented a VTQA model that combines visual and paragraph-captioning features to significantly improve visual question answering accuracy, via a model that performs early, late, and later fusion. While our model showed promising results, it still used a pre-trained paragraph captioning model to obtain the textual symbolic information. In future work, we are investigating whether the VTQA model can be jointly trained with the paragraph captioning model.

# CHAPTER 4: SEMI-SYMBOLIC MATCHING FOR VIDEO QUESTION ANSWERING

## 4.1 Introduction

Recent years have witnessed a paradigm shift in the way we get our information, and a lot of it is related to watching and listening to videos that are shared in huge amounts via the internet and new high-speed networks. Videos convey a diverse breadth of rich information, such as dynamic spatio-temporal relationships between people/objects, as well as events. Hence, it has become important to develop automated models that can accurately extract such precise multimodal information from videos (Tapaswi et al., 2016; Maharaj et al., 2017; Kim et al., 2017; Jang et al., 2017; Gao et al., 2017; Anne Hendricks et al., 2017; Lei et al., 2018, 2020b). Video question answering is a representative AI task through which we can evaluate such abilities of an AI agent to understand, retrieve, and return desired information from given video clips.

In this paper, we propose a model that effectively integrates multimodal information and locates the relevant frames from diverse, complex video clips such as those from the video+dialogue TVQA dataset (Lei et al., 2018), which contains questions that need both the video and the subtitles to answer. When given a video clip and a natural language question based on the video, naturally, the first step is to compare the question with the content (objects and keywords) of the video frames and subtitles, then combine information from different video frames and subtitles to answer the question. Analogous to this process, we apply dual-level attention in which a question and video/subtitle are aligned in word/object level, and then the aligned features from video and subtitle respectively are aligned the second time at the frame-level to integrate information for answering the question. Among the aligned frames (which contain aggregated video and subtitle information now), only those which contain relevant information for answering the question are needed. Hence, we also apply gating mechanisms to each frame feature to select the most informative frames before feeding them to the classifier.

Next, in order to make the frame selection more effective, we cast the frame selection subtask as a multi-label classification task. To convert the time span annotation to the label for each frame, we assign a positive label ('1') to frames between the start and end points, and negative ('0') label to the others, then train them with the binary cross-entropy loss. Moreover, for enhanced supervision from the human importance annotation, we also introduce a new loss function, In-and-Out Frame Score Margin (IOFSM), which is the difference in average scores between in-frames (which are inside the time span) and out-frames (which are outside the time span). We empirically show that these two losses are complementary when they are used together. Also, we introduce a way of applying binary cross-entropy to the unbalanced dataset. As we see each frame as a training example (positive or negative), we have a more significant number of negative examples than positive ones. To balance the bias, we calculate normalized scores by averaging the loss separately for each label. This modification, which we call balanced binary cross-entropy (BBCE), helps adjust the imbalance and further improve the performance of our model.

Finally, we also employ dense captions to help further improve the temporal localization of our video-QA model. Captions have proven to be helpful for vision-language tasks (Wu et al., 2019; Li et al., 2019; Kim and Bansal, 2019) by providing additional, complementary information to the primary task in descriptive textual format. We employ dense captions as an extra input to our model since dense captions describe the diverse salient regions of an image in object-level detail, and hence they would give more useful clues for question answering than single, non-dense image captions.

Empirically, our first basic model (with dual-level attention and frame-selection gates) outperforms the state-of-the-art models on TVQA validation dataset (72.53% as compared to 71.13% previous state-of-the-art) and with the additional supervision via the two new loss functions and the employment of dense captions, our model gives further improved results (73.34% and 74.20%

respectively). These improvements from each of our model components (i.e., new loss functions, dense captions) are statistically significant. Overall, our full model's test-public score substantially outperforms the state-of-the-art score by a large margin of 3.57% (74.09% as compared to 70.52%).[1] Also, our model's scores across all the 6 TV shows are more balanced than other models in the TVQA leaderboard[2], implying that our model should be more consistent and robust over different genres/domains that might have different characteristics from each other.

Our contributions are four-fold: (1) we present an effective model architecture for the video question answering task using dual-level attention and gates which fuse and select useful spatial-temporal information, (2) we employ dense captions as salient-region information and integrate it into a joint model to enhance the videoQA performance by locating proper information both spatially and temporally in rich textual semi-symbolic format, (3) we cast the frame selection sub-task as a multi-level classification task and introduce two new loss functions (IOFSM and BBCE) for enhanced supervision from human importance annotations (which could be also useful in other multi-label classification settings), and (4) our model's score on the test-public dataset is 74.09%, which is around 3.6% higher than the state-of-the-art result on the TVQA leaderboard (and our model's scores are more balanced/consistent across the diverse TV show genres). We also present several ablation and visualization analyses of our model components (e.g., the word/object-level and the frame-level attention).

### 4.2 Model

Our model consists of 2 parts: feature fusion and frame selection. For feature fusion, we introduce dual-level (word/object and frame level) attention, and we design the frame selection problem as a multi-label classification task and introduce 2 new loss functions for enhanced supervision (Figure 4.1).

[1] At the time of the ACL2020 submission deadline, the publicly visible rank-1 entry was 70.52%. Since then, there are some new entries, with results up to 71.48% (compared to our 74.09%).

[2] https://competitions.codalab.org/competitions/20415#results

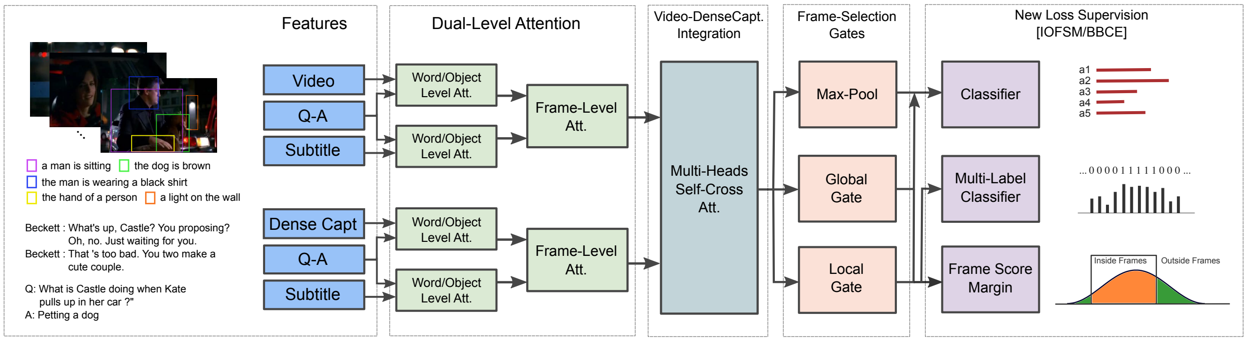


Figure 4.1: Our model consists of three parts: Dual-Level Attention, Video-DenseCapt Integration, and Frame-Selection Gates. The new loss functions (IOFSM/BBCE) also help improve the model with enhanced supervision.

### 4.2.1 Features

We follow the same approach of Lei et al. (2020b)'s work to obtain features from video, question-answer pairs, and subtitle input and encode them. We sample frames at 0.5 fps and extract object features from each frame via Faster R-CNN (Girshick, 2015). Then we use PCA to get features of 300 dimension from top-20 object proposals. We also create five hypotheses by concatenating a question feature with each of five answer features, and we pair each visual frame feature with temporally neighboring subtitles. We encode all the features using convolutional encoder.

$$\phi_{en}(x):\left\{\begin{aligned} x_0^0 &= E_{pos}(x) \\ x_t^i &= f_{i,t}(x_{t-1}^i) + x_{t-1}^i, \\ f_i(x_0^i) &= g_n(x_L^i) \\ y &= f_N \circ ... \circ f_1(x_0^0) \end{aligned}\right. \tag{4.1}$$

where $E_{pos}$ denotes positional encoding, $f_{i,t}$ convolution preceded by Layer Normalization and followed by ReLU activation, and $g_n$ the layer normalization. The encoder is composed of $N$ blocks iterations. In each iteration, the encoded inputs are transformed $L$ times of convolutions. The $L$ is set to 2, and $N$ to 1 in our experiment (Figure 4.2).

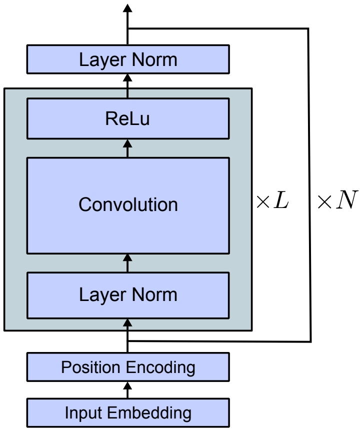


Figure 4.2: CNN encoder. We use this block to encode all the input features.

### 4.2.2 Dual-Level Attention

In dual-level attention, features are sequentially aligned in word/object-level and frame-level (Figure 4.3).

**Word/Object-Level Attention** The QA feature, $qa = \{qa_0, qa_1, .., qa_{T_{qa}}\}$, are combined with subtitle feature, $s_t = \{s_{t0}, s_{t1}, .., s_{tT_{s_t}}\}$, and visual feature, $v_t = \{v_{t0}, v_{t1}, .., v_{tT_{v_t}}\}$, of $t$-th frame respectively via word/object-level attention. To be specific, we calculate similarity matrices following Seo et al. (2017)'s approach, $S_t^v \in \mathbb{R}^{T_{qa} \times T_{s_t}}$ and $S_t^s \in \mathbb{R}^{T_{qa} \times T_{v_t}}$, from QA/subtitle and QA/visual features respectively. From the similarity matrices, attended subtitle features are obtained and combined with the QA features by concatenating and applying a transforming function. Then, max-pooling operation is applied word-wise to reduce the dimension.

$$(S_t^s)_{ij} = qa_i^\top s_{tj} \tag{4.2}$$

$$s_t^{att} = \text{softmax}(S_t^s) \cdot s_t \tag{4.3}$$

$$qa_s^m = \text{maxpool}(f_1([qa; s_t^{att}; qa \odot s_t^{att}])) \tag{4.4}$$

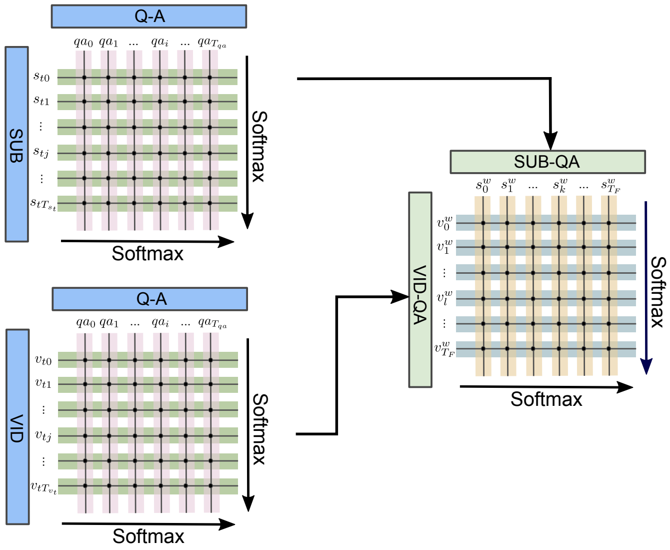


Figure 4.3: Dual-Level Attention. Our model performs two-level attentions (word/object and frame level) sequentially. In the word/object-level attention, each word/object is aligned to relevant words or objects. In the frame-level attention, each frame (which has integrated information from the word/object-level attention) is aligned to relevant frames.

where $f_1$ is a fully-connected layer followed by ReLU non-linearity. The same process is applied to the QA features.

$$qa^{att} = \text{softmax}(S_t^{s\top}) \cdot qa \tag{4.5}$$

$$s_t^m = \text{maxpool}(f_1([s_t; qa^{att}; s_t \odot qa^{att}])) \tag{4.6}$$

The fused features from different directions are integrated by concatenating and being fed to a function as follows:

$$s_t^w = f_2([qa_s^m; s_t^m; qa_s^m \odot s_t^m; qa_s^m + s_t^m]) \tag{4.7}$$

where $f_2$ is the same function as $f_1$ with non-shared parameters. All this process is also applied to visual features to get word/object-level attended features.

$$v_t^w = f_2([qa_v^m; v_t^m; qa_v^m \odot v_t^m; qa_v^m + v_t^m]) \tag{4.8}$$

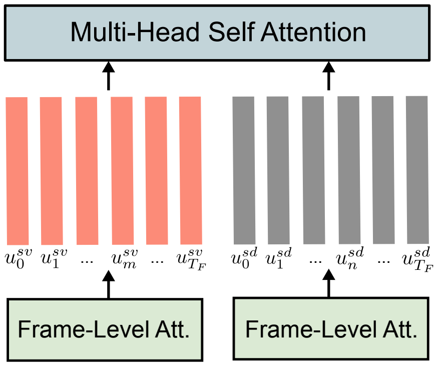

Multi-Head Self Attention
$u_0^{sv}$ $u_1^{sv}$ ... $u_m^{sv}$ ... $u_{T_F}^{sv}$
$u_0^{sd}$ $u_1^{sd}$ ... $u_n^{sd}$ ... $u_{T_F}^{sd}$
Frame-Level Att.
Frame-Level Att.

Figure 4.4: Self-Cross Attention. We combine information each from the video (fused with subtitle and QA) and dense caption (fused with subtitle and QA) via the multi-head self attention. Before being fed to the multi-head self attention module, video and dense caption features are concatenated. Thus, self and cross attentions are performed simultaneously.

**Frame-Level Attention** The fused features from word/object-level attention are integrated frame-wise via frame-level attention. Similar to the word/object-level attention, a similarity matrix, $S \in \mathbb{R}^{T_F \times T_F}$, is calculated, where $T_F$ is the number of frames. Also, from the similarity matrix, attended frame-level features are calculated.

$$(S)_{kl} = s_k^{w\top} v_l^w \tag{4.9}$$

$$s^{att} = \text{softmax}(S) \cdot s^w + s^w \tag{4.10}$$

$$\hat{v} = f_3([v^w; s^{att}; v^w \odot s^{att}; v^w + s^{att}]) \tag{4.11}$$

$$v^{att} = \text{softmax}(S^\top) \cdot v^w + v^w \tag{4.12}$$

$$\hat{s} = f_3([s^w; v^{att}; s^w \odot v^{att}; s^w + v^{att}]) \tag{4.13}$$

where $f_3$ is the same function as $f_1$ and $f_2$ with non-shared parameters. The frame-wise attended features are added to get an integrated feature.

$$u^{sv} = \hat{s} + \hat{v} \tag{4.14}$$

### 4.2.3 Video and Dense Caption Integration

We also employ dense captions to help further improve the temporal localization of our video-QA model. They provide more diverse salient regional information (than the usual single non-dense image captions) about object-level details of image frames in a video clip, and also allow the model to explicitly (in textual/semi-symbolic form) match keywords/patterns between dense captions and questions to find relevant locations/frames.

We apply the same procedure to the dense caption feature by substituting video features with dense caption features to obtain $u^{sd}$. To integrate $u^{sv}$ and $u^{sd}$, we employ multi-head self attention (Figure 4.4). To be specific, we concatenate $u^{sv}$ and $u^{sd}$ frame-wise then feed them to the self attention function.

$$\phi_{\text{self-att}}(x) \begin{cases} h_{\text{i}} = g_a(w_q^\top x_i, w_k^\top x_i, w_v^\top x_i) \\ y = w_m^\top [h_1; \ldots; h_{\text{k}}] \end{cases} \tag{4.15}$$

where $g_a$ denotes self-attention.

$$u^{svd} = \phi_{\text{self-att}}([u^{sv}; u^{sd}]) \tag{4.16}$$

In this way, $u^{sv}$ and $u^{sd}$ attend to themselves while attending to each other simultaneously. We split the output, $u^{svd}$ into the same shape as the input, then add the two.

$$z = u^{svd}[0 : T_F] + u^{svd}[T_F : 2T_F] \tag{4.17}$$

### 4.2.4 Frame-Selection Gates

To select appropriate information from the frame-length features, we employ max-pooling and gates. Features from the video-dense caption integration are fed to the CNN encoder. A fully-connected layer and sigmoid function are applied sequentially to the output feature to get frame scores that indicate how relevant each frame is for answering a given question. We get weighted features by multiplying the output feature from the CNN encoder with the scores.

$$\hat{z} = \phi_{\text{en2}}(z) \tag{4.18}$$

$$g^L = \text{sigmoid}(f^L(\hat{z})) \tag{4.19}$$

$$z^{gl} = \hat{z} \odot g^L \tag{4.20}$$

We calculate another frame scores with a different function $f^G$ to get another weighted feature.

$$g^G = \text{sigmoid}(f^G(\hat{z})) \tag{4.21}$$

$$z^{gg} = \hat{z} \odot g^G \tag{4.22}$$

Finally, following Lei et al. (2020b)'s work, we also apply frame-wise max-pooling.

$$z^{max} = \text{maxpool}(\hat{z}) \tag{4.23}$$

The three features (from local gate, global gate, and max-pooling, respectively), are then concatenated and fed to the classifier to give scores for each candidate answer.

$$logit = \text{clssifier}([z^{max}; z^{gg}; z^{gl}]) \tag{4.24}$$

We get the logits for the five candidate answers and choose the highest value as the predicted answer.

$$loss_{cls} = -\text{log}(\frac{e^{s_g}}{\sum_k e^{s_k}}) \tag{4.25}$$

where $s_g$ is the logit of ground-truth answer.

### 4.2.5 Novel Frame-Selection Supervision Loss Functions

We cast frame selection as a multi-label classification task. The frame scores from the local gate, $g^L$, are supervised by human importance annotations, which are time spans (start-end points pair) annotators think needed for selecting correct answers. To this end, we transform the time span into ground-truth frame scores, i.e., if a frame is within the time span, the frame has '1' as its label and a frame outside the span gets '0'. In this way, we can assign a label to each frame, and frames should get as close scores as their ground-truth labels. We train the local gate network with binary cross-entropy (BCE) loss.

$$loss_{bce} = -\sum_{i}^{T_F}(y\text{log}(s_i^f) + (1-y)\text{log}(1-s_i^f)) \tag{4.26}$$

where $s_i^f$ is a frame score of $i$-th frame, and y is a corresponding ground-truth label.

**In-and-Out Frame Score Margin** For additional supervision other than the binary cross-entropy loss, we create a novel loss function, In-and-Out Frame Score Margin (IOFSM).

$$loss_{io} = 1 + \text{Avg}(\text{OFS}) - \text{Avg}(\text{IFS}) \tag{4.27}$$

where OFS (Out Frame Score) is scores of frames whose labels are '0' and IFS (In Frame Score) is scores of frames whose labels are '1'.

**Balanced Binary Cross-Entropy** In our multi-label classification setting, each frame can be considered as one training example. Thus, the total number of examples and the proportion between positive and negative examples vary for every instance. This variation can cause unbalanced training since negative examples usually dominate. To balance the unbalanced training, we apply a simple but effective modification to the original BCE, and we call it Balanced Binary Cross-Entropy (BBCE). To be specific, instead of summing or averaging through the entire frame examples, we divide the positive and negative examples and calculate the average cross-entropy scores separately, then sum them together.

$$
\begin{aligned}
loss_{bbce} = -\Big( \sum_{i}^{T_{F_{in}}} \log(s_i^{f_{in}})/T_{F_{in}} \\
+ \sum_{j}^{T_{F_{out}}} \log(1 - s_j^{f_{out}})/T_{F_{out}} \Big)
\end{aligned}
\tag{4.28}
$$

where $s_i^{f_{in}}$ and $s_j^{f_{out}}$ are $i$-th in-frame score and $j$-th out-frame score respectively, and $T_{F_{in}}$ and $T_{F_{out}}$ are the number of in-frames and out-frames respectively.

Thus, the total loss is:

$$
loss = loss_{cls} + loss_{(b)bce} + loss_{io} \tag{4.29}
$$

## 4.3 Experimental Setup

**TVQA Dataset** TVQA dataset (Lei et al., 2018) consists of video frames, subtitles, and question-answer pairs from 6 TV shows. The number of examples for train/validation/test-public dataset are 122,039/15,253/7,623. Each example has five candidate answers with one of them the ground-truth. So, TVQA is a classification task, in which models select one from the five candidate answers, and models can be evaluated on the accuracy metric.

**Dense Captions** We use Yang et al. (2017)'s pretrained model to extract dense captions from each video frame. We extract the dense captions in advance and use them as extra input data to the model.[3]

**Training Details** We use GloVe (Pennington et al., 2014) word vectors with dimension size of 300 and RoBERTa (Liu et al., 2019) with 768 dimension. The dimension of the visual feature is 300, and the base hidden size of the whole model is 128. We use Adam (Kingma and Ba, 2015)

[3] This is less computationally expensive and dense captions from the separately trained model will be less biased towards the questions of TVQA dataset, and hence provide more diverse aspects of image frames of a video clip.

|  | Model | Test-Public (%) |  |  |  |  |  |  | Val (%) |
|---|---|---|---|---|---|---|---|---|---|
|  |  | all | bbt | friends | himym | grey | house | castle |  |
| 1 | jacobssy (anonymous) | 66.01 | 68.75 | 64.98 | 65.08 | 69.22 | 66.45 | 63.74 | 64.90 |
| 2 | multi-stream (Lei et al., 2018) | 66.46 | 70.25 | 65.78 | 64.02 | 67.20 | 66.84 | 63.96 | 65.85 |
| 3 | PAMN (Kim et al., 2019b) | 66.77 | - | - | - | - | - | - | 66.38 |
| 4 | Multi-task (Kim et al., 2019a) | 67.05 | - | - | - | - | - | - | 66.22 |
| 5 | ZGF (anonymous) | 68.77 | - | - | - | - | - | - | 68.90 |
| 6 | STAGE (Lei et al., 2020b) | 70.23 | - | - | - | - | - | - | 70.50 |
| 7 | akalsdnr (anonymous) | 70.52 | 71.49 | 67.43 | 72.22 | 70.42 | 70.83 | 72.30 | 71.13 |
| 8 | Ours (hstar) | **74.09** | **74.04** | **73.03** | **74.34** | **73.44** | **74.68** | **74.86** | **74.20** |

Table 4.1: Our model outperforms the state-of-the-art models by a large margin. Moreover, the scores of our model across all the TV shows are more balanced than the scores from other models, which means our model is more consistent/robust and not biased to the dataset from specific TV shows.

as the optimizer. We set the initial learning rate to 0.001 and drop it to 0.0002 after running 10 epochs. For dropout, we use the probability of 0.1.

### 4.4 Results and Ablation Analysis

As seen from Table 4.1, our model outperforms the state-of-the-art models in the TVQA leaderboard. Especially our model gets balanced scores for all the TV shows while some other models have high variances across the shows. As seen from Table 4.2, the standard deviation and 'max-min' value over our model's scores for each TV show are 0.65 and 1.83, respectively, which are the lowest values among all models in the list. This low variance could mean that our model is more consistent and robust across all the TV shows.

**Model Ablations** As shown in Table 4.3, our basic dual-attention and frame selection gates model shows substantial improvement over the strong single attention and frame span baseline (row 4 vs 1: $p < 0.0001$), which is from the best published model (Lei et al., 2020b). Each of our dual-attention and frame selection gates alone shows a small improvement in performance than the baseline (row 3 vs 1 and 2 vs 1, respectively).[4] However, when they are applied together,

[4] Although the improvements are not much, but performing word/object level attention and then frame level attention is more intuitive and interpretable than a non-dual-attention method, allowing us to show how the model works: see visualization in Sec. 4.5.

| | Model | TV Show Score | | |
|---|---|---|---|---|
| | | avg. | std. | max-min |
| 1 | jacobssy (anonymous) | 66.37 | 2.01 | 5.48 |
| 2 | multi-stream (Lei et al., 2018) | 66.34 | 2.15 | 6.29 |
| 3 | akalsdnr (anonymous) | 70.78 | 1.65 | 4.87 |
| 4 | Ours | 74.07 | 0.65 | 1.83 |

Table 4.2: Average and standard deviation of the test-public scores from each TV show (for this comparison, we only consider models that release the scores for each TV show).[6]

| | Model | Val Score (%) |
|---|---|---|
| 1 | Single-Att + Frame-Span | 69.86 |
| 2 | Single-Att + Frame-Selection Gates | 70.08 |
| 3 | Dual-Att + Frame-Span | 70.20 |
| 4 | Dual-Att + Frame-Selection Gates (w/o NewLoss) | 71.26 |
| 5 | Dual-Att + Frame-Selection Gates | 72.51 |
| 6 | Dual-Att + Frame-Selection Gates (w/o NewLoss) + RoBERTa | 72.53 |
| 7 | Dual-Att + Frame-Selection Gates + RoBERTa | 73.34 |
| 8 | Dual-Att + Frame-Selection Gates + RoBERTa + DenseCapts | 74.20 |

Table 4.3: Model Ablation: our dual-attention / frame-selection Gates, new loss functions, and dense captions help improve the model's performance (NewLoss: IOFSM+BBCE).

the model works much better. The reason why they are more effective when put together is that frame selection gates basically select frames based on useful information from each frame feature and our dual-attention can help this selection by getting more relevant information to each frame through the frame-level attention. Next, our new loss functions significantly help over the dual-attention and frame selection gates model by providing enhanced supervision (row 5 vs 4: $p < 0.0001$, row 7 vs 6: $p < 0.005$). Our RoBERTa version is also significantly better than the GloVe model (row 6 vs 4: $p < 0.0005$, row 7 vs 5: $p < 0.01$). Finally, employing dense captions further improves the performance via useful textual clue/keyword matching (row 8 vs 7: $p < 0.005$).[5]

**IOFSM and BCE Loss Functions Ablation and Analysis** To see how In-and-Out Frame Score Margin (IOFSM) and Binary Cross-Entropy (BCE) loss affect the frame selection task, we compare the model's performance/behaviors according to the combination of IOFSM and

[5] Statistical significance is computed using the bootstrap test (Efron and Tibshirani, 1994).

|  | Loss | Val Score (%) | IFS |  | OFS |  |
|---|---|---|---|---|---|---|
|  |  |  | avg | std | avg | std |
| 1 | BCE | 71.26 | 0.468 | 0.108 | 0.103 | 0.120 |
| 2 | IOFSM | 70.75 | 0.739 | 0.127 | 0.143 | 0.298 |
| 3 | BCE+IOFSM | 72.22 | 0.593 | 0.128 | 0.111 | 0.159 |
| 4 | BBCE | 72.27 | 0.759 | 0.089 | 0.230 | 0.231 |
| 5 | BBCE+IOFSM | 72.51 | 0.764 | 0.098 | 0.182 | 0.246 |

Table 4.4: IOFSM and BBCE help improve the model's performance by changing in and out-frame scores.

BCE. As shown in Table 4.4, applying IOFSM on top of BCE gives a better result. When we compare row 1 and 3 in Table 4.4, the average in-frame score of BCE+IOFSM is higher than BCE's while the average out-frame scores of both are almost the same. This can mean two things: (1) IOFSM helps increase the scores of in-frames, and (2) increased in-frame scores help improve the model's performance. On the other hand, when we compare row 1 and 2, the average in-frame score of IOFSM is higher than BCE's. But, the average out-frame score of IOFSM is also much higher than BCE's. This can mean that out-frame scores have a large impact on the performance as well as in-frame scores. This is intuitively reasonable. Because information from out-frames also flows to the next layer (i.e., classifier) after being multiplied by the frame scores, the score for the 'negative' label also has a direct impact on the performance. So, making the scores as small as possible is also important. Also, when we compare the row 2 and others (2 vs. 1 and 3), the gap between in-frame scores is much larger than the gap between out-frame scores. But, considering the scores are average values, and the number of out-frames is usually much larger than in-frames, the difference between out-frame scores would affect more than the gap itself.

**Balanced BCE Analysis** We can see from row 1 and 4 of the Table 4.4 that BBCE shift the average scores of both in-frames and out-frames to higher values. This can show that scores from the BCE loss are biased to the negative examples, and BBCE can adjust the bias with the separate averaging. The score shift can help improve the model's performance. But, when comparing row 2 and 4, the out-frame scores of BBCE are higher than IOFSM, and this may imply that the result from BBCE should be worse than IOFSM since out-frame scores have a large impact on the performance. However, as we can see from row 2, the standard deviation of IOFSM's out-frame scores is larger than BBCE. This could mean that a model with IOFSM has an unstable scoring behavior, and it could affect the performance. As seen from row 5, applying BBCE and IOFSM together gives further improvement, possibly due to the increased in-frame scores and decreased out-frame scores while staying around at a similar standard deviation value.

### 4.5 Visualizations

In this section, we visualize the dual-level attention (word/object and frame level) and the frame score change by new losses application (for all these attention examples, our model predicts the correct answers).

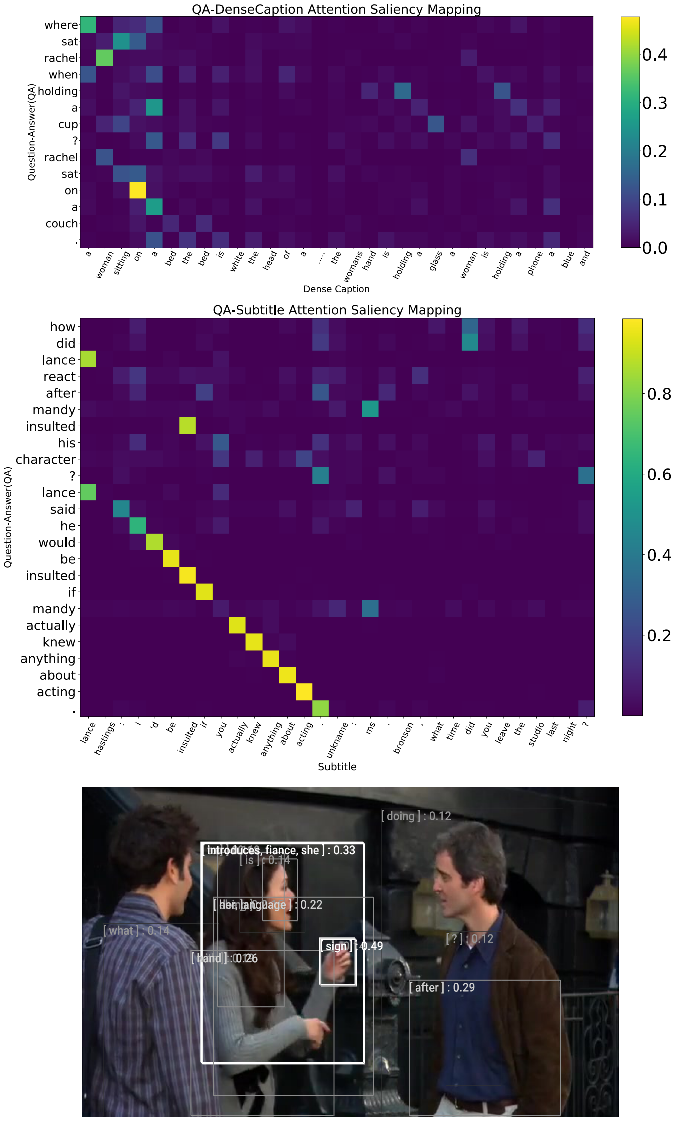
**Word/Object-Level Attention** We visualize word-level attention in Figure 4.5. In the top example, the question and answer pair is "Where sat Rachel when holding a cup?" - "Rachel sat on a couch". Our word/object-level attention between QA pair and dense caption attend to a relevant description like 'holding a glass' to help answer the question. In the middle example, the question and answer pair is, "How did Lance react after Mandy insulted his character?" - "Lance said he would be insulted if Mandy actually knew anything about acting". Our word/object-level attention between QA pair and subtitle properly attend to the most relevant words such as 'insulted', 'knew', and 'acting' to answer the question. In the bottom example, the question and answer pair is, "What is Cathy doing with her hand after she introduces her fiance to Ted?" - "She is doing sign language". From the score of our word/object-level attention, the model aligns the word 'sign' to the woman's hand to answer the question.

**Frame-Level Attention** As shown in Figure 4.6, our frame-level attention can align relevant frames from different features. In the example, the question and answer pair is "Where did Esposito search after he searched Carol's house downstairs?" - "Upstairs". To answer this question, the model needs to find a frame in which 'he (Esposito) searched Carol's house downstairs', then find a frame which has a clue for ‘where did Esposito search’. Our frame-level attention can properly align the information fragments from different features (Frame 20 and 25) to help answer questions.

**Frame Score Enhancement by New Losses** As seen in Figure 4.7, applying our new losses (IOFSM+BBCE) changes the score distribution over all frames. Before applying our losses (left figure), overall scores are relatively low. After using the losses, overall scores increased, and especially, scores around in-frames get much higher.

### 4.6 Conclusion

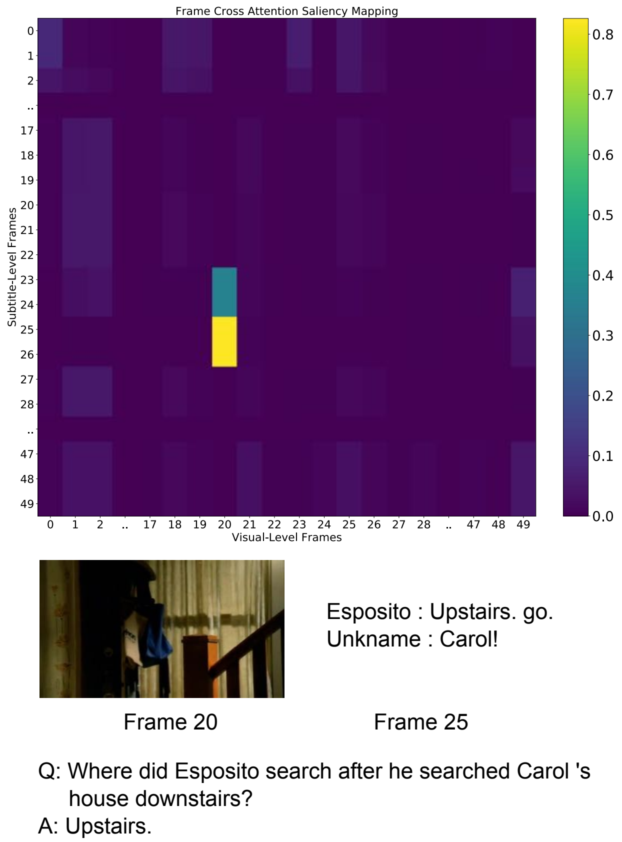
We presented our dual-level attention and frame-selection gates model and novel losses for more effective frame-selection. Furthermore, we employed dense captions to help the model better find clues from salient regions for answering questions. Each component added to our base model architecture (proposed loss functions and the adoption of dense captions) significantly improves the model’s performance. Overall, our model outperforms the state-of-the-art models on the TVQA leaderboard, while showing more balanced scores on the diverse TV show genres.



Q: What is Cathy doing with her hand after she introduces her fiance to Ted?

A: She is doing sign language.

Figure 4.5: Visualization of word/object level attention. Top: words from a question-answer pair to words from dense captions alignment. Middle: words from a question-answer pair to words from subtitles alignment. Bottom: words from a question-answer pair to regions (boxes) from an image (only boxes with top 1 scores from each word are shown).



Esposito : Upstairs. go.
Unkname : Carol!

Q: Where did Esposito search after he searched Carol 's house downstairs?
A: Upstairs.

Figure 4.6: Visualization of frame-level attention. Frame 25 (which contains 'upstairs') from subtitle features and frame 20 (which shows 'downstairs' by banister upward) from visual features are aligned.

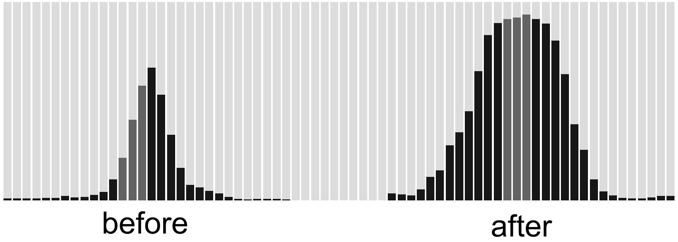
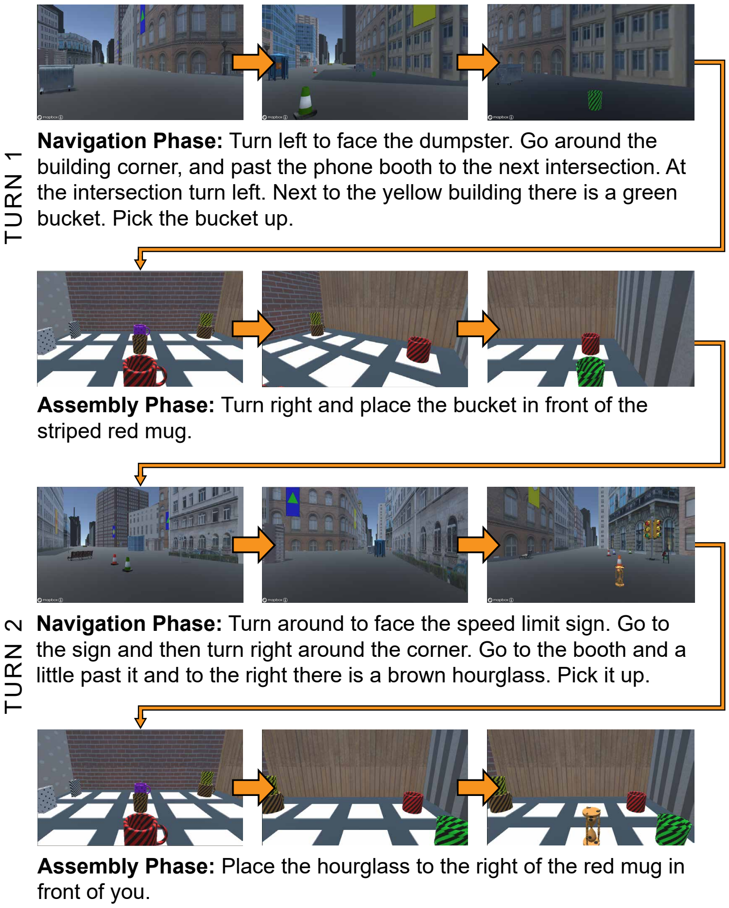


Figure 4.7: Visualization of distribution change in frame selection scores. Left: the score distribution before applying new losses (IOFSM+BBEC). Right: the score distribution after applying the losses. Scores neighboring in-frame (gray) are increased. For this example, the model does not predict the right answer before applying the losses, but after training with the losses, the model chooses the correct answer.

# CHAPTER 5: INSTRUCTION FOLLOWING EMBODIED NAVIGATION AND MANIPULATION

## 5.1 Introduction

Navigation guided via flexible natural language (NL) instructions is a crucial capability for robotic and embodied agents. Such systems should be capable of interpreting human instructions to correctly navigate realistic complex environments and reach destinations by understanding the environment, and associating referring expressions in the instructions with the corresponding visual cues in the environment. Many research efforts have focused on this important vision-and-language navigation task (MacMahon et al., 2006; Mooney, 2008; Chen and Mooney, 2011; Tellex et al., 2011; Mei et al., 2016; Hermann et al., 2017; Brahmbhatt and Hays, 2017; Mirowski et al., 2018; Anderson et al., 2018b; Misra et al., 2018; Blukis et al., 2018; Das et al., 2018; Cirik et al., 2018; de Vries et al., 2018; Blukis et al., 2019; Thomason et al., 2019; Nguyen et al., 2019; Nguyen and Daumé III, 2019; Chen et al., 2019; Jain et al., 2019; Shridhar et al., 2020; Qi et al., 2020; Hermann et al., 2020; Berg et al., 2020; Zhu et al., 2020a). However, in real-world applications, navigation alone is rarely the exclusive goal. In most cases, agents will navigate to perform another task at their destination, and also repeat subtasks, e.g., a warehouse robot may be asked to pick up several objects from different locations and then assemble the objects into a desired arrangement. When these additional tasks are interweaved with navigation, the degree of complexity increases exponentially due to cascading errors. Relatively few studies have focused on this idea of combining navigation with other tasks. Touchdown (Chen et al., 2019) combines navigation and object referring expression resolution, REVERIE (Qi et al., 2020) performs remote referring expression comprehension, and ALFRED (Shridhar et al., 2020) combines indoor navigation and household manipulation. However, there has been no task that integrates the navi-



**Navigation Phase:** Turn left to face the dumpster. Go around the building corner, and past the phone booth to the next intersection. At the intersection turn left. Next to the yellow building there is a green bucket. Pick the bucket up.

**Assembly Phase:** Turn right and place the bucket in front of the striped red mug.

**Navigation Phase:** Turn around to face the speed limit sign. Go to the sign and then turn right around the corner. Go to the booth and a little past it and to the right there is a brown hourglass. Pick it up.

**Assembly Phase:** Place the hourglass to the right of the red mug in front of you.

Figure 5.1: Navigation and assembly phases (2 turns), via NL (English) instructions in a dynamic 3D environment. In the navigation phase, agents are asked to find and collect a target object. In the assembly phase, agents have to egocentrically place the collected object at a relative location (navigation turn 2 starts where turn 1 ends; we only show 3 snapshots here for space reasons, but the full simulator and its image set will be made available).

gation task in complex outdoor spaces with the assembling task (and object referring expression comprehension), requiring spatial relation understanding in an interweaved temporal way, in which the two tasks alternate for multiple turns with cascading error effects (see Figure 5.1).

Thus, we introduce a new task that combines the navigation, assembling, and referring expression comprehension subtasks. This new task can be explained as an intuitive combination of the navigation and collection aspects of PokéMON GO[1] and an ARRAnging (assembling) aspect, hence we call it 'ARRAMON'. In this task, an agent needs to follow navigational NL instructions to navigate through a complex outdoor and fine-grained city environment to collect diverse tar-

[1] https://www.pokemongo.com

get objects via referring expression comprehension and dynamic 3D visuospatial relationship understanding w.r.t. other distracter objects. Next, the agent is asked to place those objects at specific locations (relative to other objects) in a grid environment based on an assembling NL instruction. These two phases are performed repeatedly in an interweaved manner to create an overall configuration of the set of collected objects. For enabling the ARRAMON task, we also implement a simulator built in the Unity game engine[2] to collect the dataset (see Appendix A.3.2 for the simulator interface). This simulator features a 3D synthetic city environment based on real-world street layouts with realistic buildings and textures (backed by Mapbox[3]) and a dynamic grid floor assembly room (Figure 5.1), both from an egocentric view (the full simulator and its image set will be made available). We take 7 disjoint sub-sections from the city map and collect instructions from workers within each section. Workers had to write instructions based on ground truth trajectories (represented as path lines in navigation, location highlighting during assembly). We placed diverse background objects as well as target objects so that the rich collected instructions require agents to utilize strong linguistic understanding. The instructions were next executed by a new set of annotators in a second verification stage and were filtered based on low match w.r.t. the original ground truth trajectory, and the accuracy of assembly placement. Overall, this resulted in a dataset of 7,692 task instances with multiple phases and turns (a total of 30,768 instructions and paths).[4] We have since extended our dataset by also collecting the corresponding Hindi instructions.

To evaluate performance in our ARRAMON task, we employ both the existing metric of nDTW (Normalized Dynamic Time Warping) (Ilharco et al., 2019) and our newly-designed metrics: CTC-k (Collected Target Correctness), rPOD (Reciprocal Placed Object Distance), and PTC (Placed Target Correctness). In the navigation phase, nDTW measures how similar generated paths are to the ground truth paths, while CTC-k computes how closely agents reach the

[2] https://www.unity.com

[3] https://www.mapbox.com

[4] Our dataset size is comparable to other similar tasks (e.g., R2R, Touchdown, ALFRED, CVDN, REVERIE; we are also planning to further increase the size and add other languages.

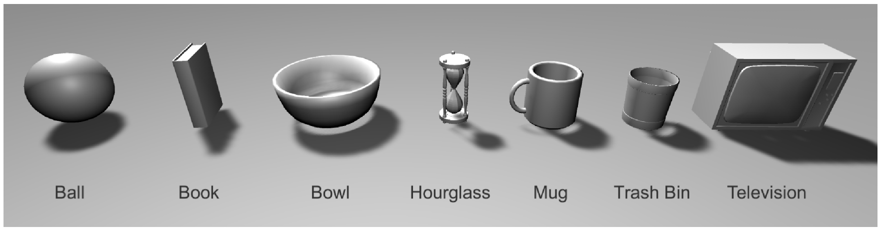


Figure 5.2: Illustration of the basic object types that the agent must collect, and will also appear as distracter objects during both navigation and assembly phases.

targets. In the assembly phase, rPOD calculates the reciprocal distance between target and agents' placement locations, and PTC counts the correspondence between those locations. Due to the interweaving property of our task with multiple navigation and assembling phases and turns, performance in the previous turn and phase cascadingly affects the metric scoring of the next turn and phase (Section 5.2.2).

Lastly, we implement multiple baselines as good starting points and to verify our task is challenging and the dataset is unbiased. We present integrated vision-and-language, vision-only, language-only, and random-walk baselines. Our vision-and-language model shows better performance over the other baselines, which implies that our ARRAMON dataset is not skewed; moreover, there exists a very large gap between this model and the human performance, implying that our ARRAMON task is challenging and that there is substantial room for improvements by future work. We will publicly release the ARRAMON simulator, dataset, and code, along with a leaderboard to encourage further community research on this realistic and challenging joint navigation-assembly task.

## 5.2 Task

The ARRAMON task consists of two phases: navigation and assembly. We define one turn as one navigation phase plus one assembly phase (see Figure 5.1). Both phases are repeated twice (i.e., 2 turns), starting with the navigation phase. During the navigation phase, an agent is asked to navigate a rich outdoor city environment by following NL instructions, and then collect the target object identified in the instructions via diverse referring expressions. During the assembly

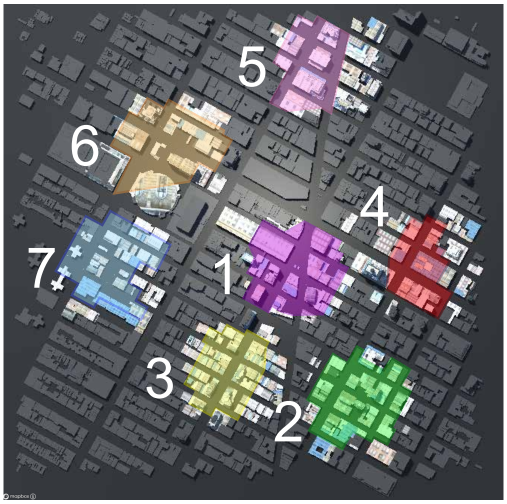


Figure 5.3: Illustration of the seven city sections in which data was collected.

phase, the agent is asked to place the collected object (from the previous navigation phase) at a target location on a grid layout, using a different NL instruction via relative spatial referring expressions. Target objects and distracter objects are selected from one of seven objects shown in Figure 5.2 and then are given one of two different patterns and one of seven different colors (see Figure A.6 in Appendices). In both phases, the agent can take 4 actions: forward, left, right, and an end pickup/place action. Forward moves the agent 1 step ahead and left/right makes agents rotate 30° in the respective direction.[5]

### 5.2.1 Environment

**Navigation Phase.** In this phase, agents are placed at a random spot in one of the seven disjoint subsections of the city environment (see Figure 5.3), provided with an NL instruction, and asked to find the target object. The city environment is filled with background objects: buildings and various objects found on streets (see Figure 5.4). There are also a few distracter objects in the city that are similar to target objects (in object type, pattern, and color). During this phase, the agent's end action is 'pick-up'. The pick-up action allows agents to pick up any collectible object within

[5] In our task environment, holistically, the configuration of the set of objects dynamically changes as agents pick-up and place or stack them relative to the other objects, which is one challenging interaction between the objects.

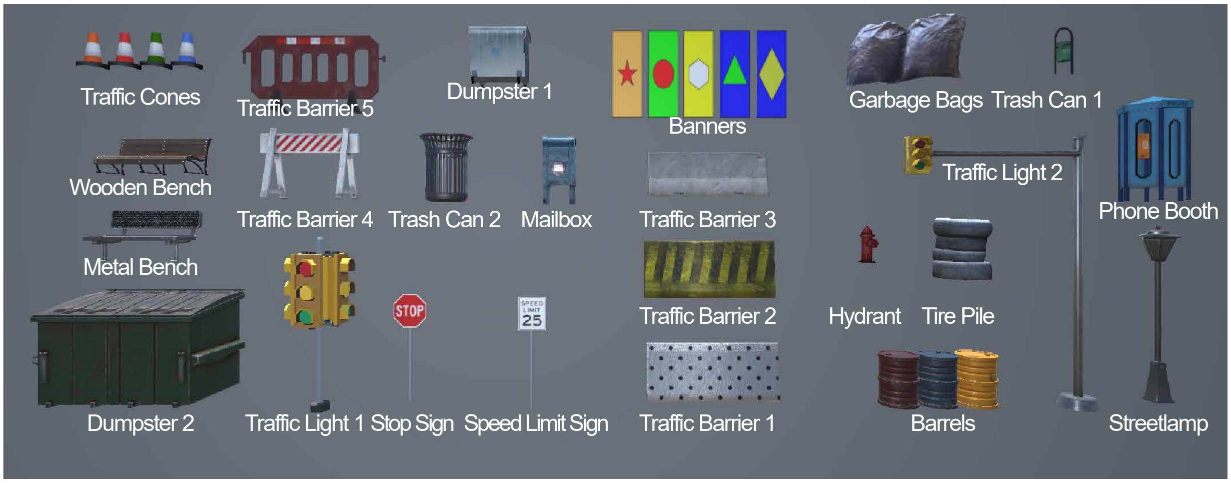


Figure 5.4: Illustration of the background environmental objects scattered around the city environment.

range (a rectangular area: 0.5 unit distance from an agent toward both their left and right hand side and 3 unit distance forward).

**Assembly Phase.** Once the agent picks up the collectible object in the navigation phase, they enter the assembly phase. In this phase, agents are again provided with an NL instruction, but they are now asked to place the target object they collected in the previous phase at the target location identified in the instruction. When the assembly phase begins, 8 decoy basic-type objects (Figure 5.2) with random pattern and color, are placed for use as distractions. In this phase, agents can only move on a 4-by-5 grid layout. The grid is bordered by 4 walls, each with a different texture/pattern (wood, brick, spotted, striped) to allow for more diverse expressions in the assembly phase. Their end action is ‘place’, which puts the collected object onto the grid one step ahead. Agents cannot place diagonally and, unlike in the navigation phase, cannot move forward diagonally.

Hence, to accomplish the overall joint navigation-assembly task, it is required for agents to have integrated abilities. During navigation they must take actions based on understanding the egocentric view and aligning the NL instructions with the dynamic visual environment to successfully find the target objects (relevant metrics: nDTW and CTC-k, see Section 5.2.2). During assembly, from an egocentric view, they must understand 3D spatial relations among objects identified by referring expressions in order to place the target objects at the right relative location. (relevant metrics: PTC and rPOD, see Section 5.2.2).[6]

### 5.2.2 Metrics

**Normalized Dynamic Time Warping (nDTW).** To encourage the agent to follow the paths closely during the navigation task, we employ nDTW (Ilharco et al., 2019) as our task metric. nDTW measures the similarity between a ground-truth path and a predicted trajectory of an agent, thus penalizing randomly walking around to find and pick up the target object.

**Collected Target Correctness (CTC).** An agent that understands the given NL instructions well should find and pick up a correct target object at the end of the navigation task. Therefore, we evaluate the agent's ability with CTC, which will have a value of 1 if the agent picks up a correct object, and a value of 0 if they pick up an incorrect object or do not pick up any object. Since collecting the correct object is a difficult task, we also implement the CTC-k metric. CTC-k measures the CTC score at distance k. If the agent is within k distance of the target object, then the value is 1, otherwise it is 0 (CTC-0 indicates the original CTC).

**Placed Target Correctness (PTC).** In the assembly task, placing the collected object at the exact target position is most important. The PTC metric counts the correspondence between the target location and the placed location. If the placed and target locations match, then the PTC is 1, otherwise it is 0. If the collected object is not correct, then the score is also 0.

**Reciprocal Placed Object Distance (rPOD).** We also consider the distance between the target position and the position where the collected object is eventually placed in the assembly task (Bisk et al., 2018). The distance squared is taken to penalize the agent more for placing the object far from the target position. Then 1 is added and the reciprocal is taken to normalize the final metric value: $\text{rPOD} = \frac{1}{1+D_a^2}$, where $D_a$ is the Manhattan distance between the target and

[6] We assume agents backtrack their path to go back to the warehouse for assembling, after each navigation phase (since the path is known, it can be automated and there is no additional learning task involved, and so no visuals are needed). Likewise, after the assembly phase, the agent can resume at the pick-up position by re-following the previous path. One can also imagine agents are moving with a container, in which they assemble the objects as they pick them up.

placed object positions. If the collected object is not correct, then the score is 0 (see Figure A.2 in Appendices).

Overall, our metrics reflect the interweaving property of our task. For example, if agents show poor performance in the first turn navigation phase (i.e., low nDTW and CTC-k scores), they will not obtain high scores in the continuing assembly phase (i.e., low PTC and rPOD scores), also leading to lower scores in the second turn navigation phase.

## 5.3 ARRAMON Dataset

Our ARRAMON navigation-assembly dataset is a collection of rich human-written NL (English) instructions. The navigation instructions explain how to navigate the large outdoor environments and describe which target objects to collect. The assembly instructions provide the desired target locations for placement relative to objects. Each instruction set in the dataset is accompanied by ground truth (GT) trajectories and placement locations. Data was collected from the online crowd-sourcing platform Amazon Mechanical Turk (AMT).

### 5.3.1 Data Collection

The data collection process was broken into two stages: Stage 1: Writing Instructions, and Stage 2: Following/Verifying Instructions. Within each stage, there are two phases: Navigation and Assembly (see Figure A.8 in Appendices for the interface of each stage and each phase). During the first stage's navigation phase, a crowdworker is placed in the city environment as described in Section 5.2.1 and moves along a blue navigation line (representing the GT path) that will lead them to a target object (see Appendix A.3.1 for the exact route generation details). While the worker travels this line, they write instructions describing their path (e.g., "Turn to face the building with the green triangle on a blue ... Walk past the bench to the dotted brown TV and pick it up."). Workers were bound to this navigation line to ensure that they wrote instructions only based on what they could see from the GT path. Next, the worker starts the first stage's assembly phase and is placed in a small assembly room, where they must place the object they just collected in a predetermined location (indicated by a transparent black outline of the object they just collected) and write instructions on where to place the object relative to other objects from an egocentric viewpoint (e.g., “Place the dotted brown TV in front of the striped white hourglass.”). The worker is then returned to the city environment and repeats both phases once more.

A natural way of verifying the instruction sets from Stage 1 is to have new workers follow them (Chen et al., 2019). Thus, during Stage 2 Verification, a new worker is placed in the environment encountered by the Stage 1 worker and is provided with the NL instructions that were written by that Stage 1 worker. The new worker has to follow the instructions to find the target objects in the city and place them in the correct positions in the assembly environment. Each instruction set from Stage 1 is verified by three unique crowdworkers to ensure instructions are correctly verified. Next, evaluation of the Stage 2 workers performance was done through the use of the nDTW and PTC metrics. If at least one of three different Stage 2 workers scored higher than 0.2 on nDTW in both navigation turns and had a score of 1 on PTC in both assembly turns, then the corresponding Stage 1 instruction set was considered high quality and kept in the dataset, otherwise it was discarded. The remaining dataset has a high average nDTW score of 0.66 and an even higher expert score of 0.81 (see Sec. 5.7).[7]

### 5.3.2 Data Quality Control

Instructions written by the Stage 1 workers needed to be clear and understandable. Workers were encouraged to follow certain rules and guidelines so that the resulting instruction would be of high quality and made proper use of the environment.

**Guidelines, Automated Checks, and Qualification Tests.** Detailed guidelines were put in place to help ensure that the instructions written contained as few errors as possible. Rules were shown to workers before the start of the task and active automated checks took place as the workers

[7] Workers were allowed to repeat both tasks, however they were prevented from encountering an identical map setting that already has instructions during Stage 1 and their own instructions during Stage 2.

wrote. These active checks helped prevent poor instructions (such as those including certain symbols) from being submitted, requiring workers to fix them before submitting. In the case the instruction quality was questionable, an email notification was sent (see Appendix A.3.1 for the exact guidelines and checks that were implemented, as well as details regarding the email notifications). A screening test was also required at the start of both stages to test the crowdworkers' understanding of the task. If a wrong answer was chosen, an explanation was displayed and the crowdworker was allowed to try again (see Figure A.4 and A.5 in Appendices for the screening tests). To help workers place the object in the right location during Stage 2, we use a simple placement test which they pass by placing an object at the correct place during a mock assembly phase (see Appendix A.3.1 for details).

**Worker Qualifications.** Workers completing the task were required to pass certain qualifications before they could begin. As the Stage 1 and 2 tasks require reading English instructions (Stage 1 also involves writing), we required workers be from native-speaking English countries. Workers were required to have at least 1000 approved tasks and a 95% or higher approval rating. A total of 96 unique workers for Stage 1 and 242 for Stage 2, were able to successfully complete their respective tasks.

**Worker Payment and Bonus Incentives.** We kept fair and comparable pay rates based on similar datasets (Chen et al., 2019), writing (Stage 1) had a payment (including bonuses) of $1.00. Instruction verification (Stage 2) had a payment of $0.20. See Appendix A.3.1 for details on bonus criteria, rates.

### 5.4 Data Analysis

A total of 8,546 instruction sets were collected. Each set included two pairs of navigation and assembly instructions (thus, 34,184 instructions in total). After filtering from Stage 2 results, there remained 7,692 instruction sets (30,768 instructions in total). Our dataset size is comparable to other similar tasks, e.g., Touchdown (Chen et al., 2019) contains 9.3K examples (9.3K navigation and 27.5K SDR task), R2R (Anderson et al., 2018b) has 21.5K navigation instructions,

| Linguistic Property | Navigation Frequency | Assembly Frequency | Instruction Examples |
|---|---|---|---|
| Egocentric Spatial Relation | 34% | 34% | "...Go straight so the striped green bucket with the red tv on top of it is to **your right**..." |
| Allocentric Spatial Relation | 86% | 98% | "Place the dotted yellow bucket **on the left side of** the striped brown bowl." |
| Temporal Condition | 64% | 2% | "...Continue to walk forward **until you reach an intersection**..." |
| Directional Reference | 96% | 68% | "Make **a slight left** and walk forward stopping at the intersection." |
| Sequencing | 66% | 58% | "...Go forward past the dotted yellow bucket and past the lamp post near the blue phone booth..." |
| 3D Discrete Referring Expressions | 72% | 34% | "Put the striped blue book behind the dotted red mug." |

Table 5.1: Linguistic properties and their frequencies found in within 50 randomly sampled instruction sets from the ARRAMON dataset.

REVERIE has 21.7K instructions, ALFRED (Shridhar et al., 2020) has 25.7K language directives describing 8K demonstrations, and CVDN (Thomason et al., 2019) dataset with 7.4K NDH instances and 2K navigation dialogues.

**Linguistic Properties.** As shown in Table 5.1, our instruction sets have diverse linguistic features that make our task more challenging. Our ARRAMON task requires that the agent be able to understand and distinguish between both egocentric and allocentric spatial relations, necessitating that they comprehend the relation between entities in the environment according to their location and orientation. The instructions contain many directional words and phrases which require that agents utilize strong navigational skills. Additionally, due to the large scale of the environment, temporal condition expressions are crucial for agents to navigate effectively, as they are useful for describing long-distance travel. A unique linguistic property found in our sample is 3D discrete referring expressions which utilize 3D depth to guide the agent; implying that the combined navigation and assembly task requires that agents possess a full understanding of object relations in a 3D environment.

**Dataset Statistics.** Figure 5.5 shows that the most frequently occurring words in our dataset. These words are primarily directional or spatial relations. This implies that agents should be able to understand the concept of direction and the spatial relations between objects, especially as they change with movement. Table 5.2 and Figure 5.6 show that navigation tends to have longer instructions and path lengths. Assembly occurs in a smaller environment, requiring agents to

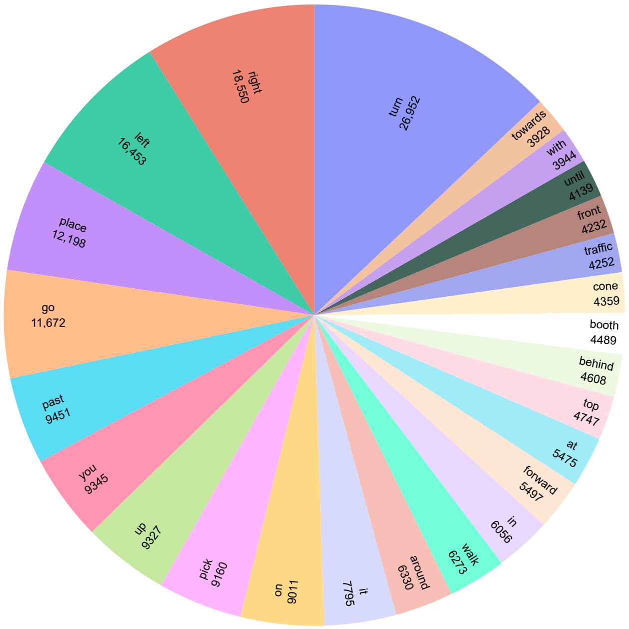


Figure 5.5: The frequency distribution of the 25 most common words in the dataset. Stopwords and target object words have been removed.

| Length | Navigation | | Assembly | |
|---|---|---|---|---|
| | max | avg. | max | avg. |
| Instruction | 147 | 47.99 | 90 | 20.99 |
| Path | 156 | 48.14 | 8 | 3.32 |
| Action Sequence | 224 | 75.78 | 34 | 13.68 |

Table 5.2: Lengths of the instructions (in words), paths, and action sequences for both turns across all subsections in the city.

focus less on understanding paths than in navigation and more on understanding the 3D spatial relations of objects from the limited egocentric viewpoint.

## 5.5 Models

We train an integrated Vision-and-Language model as a good starting point baseline for our task. To verify that our dataset is not biased towards some specific factors, we trained ablated and random walk models and evaluated them on the dataset.

**Vision-and-Language Baseline.** This model uses vision and NL instruction features together to predict the next actions (Figure 5.7). We implement each module for navigation/assembly phases

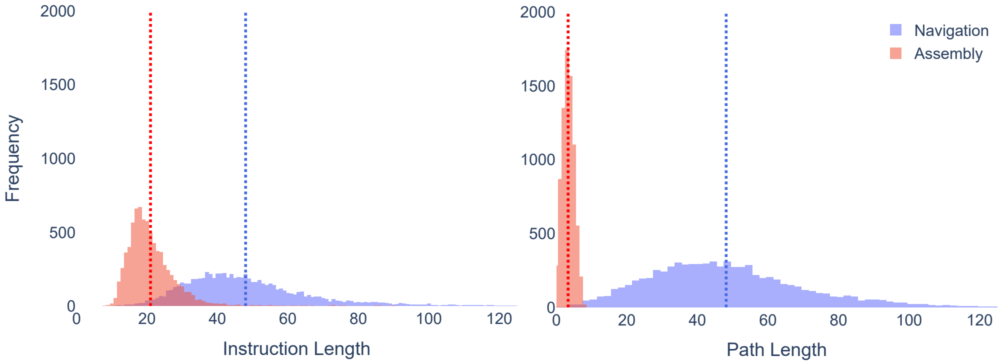


Figure 5.6: The frequency distributions of instruction lengths (left) and path lengths (right) in the navigation and assembly phases. Graphs cut off at length 125 since beyond that there are very few data points.

as:

$$L = \text{Emb}_L(\text{Inst.}), \quad \tilde{a}_t = \text{Emb}_A(a_t) \tag{5.1}$$

$$V_t = \text{Enc}_V(\text{Img}_t), \quad \tilde{L} = \text{Enc}_L(L) \tag{5.2}$$

$$h_t = \text{LSTM}(\tilde{a}_{t-1}, h_{t-1}) \tag{5.3}$$

$$\hat{V}_t, \hat{L}_t = \text{Cross-Attn}(V_t, \tilde{L}) \tag{5.4}$$

$$v_t = \text{Attn}(h_t, \hat{V}_t), \quad l_t = \text{Attn}(h_t, \hat{L}_t) \tag{5.5}$$

$$\text{logit}_{a_t} = \text{Linear}(v_t, l_t), \quad a_t = \max(\text{logit}_{a_t}) \tag{5.6}$$

where $\text{Img}_t$ is the view of an agent at time step $t$, Inst. is natural language instructions given to the agent, and $a_t$ is an action at time step $t$. Instructions and actions are embedded via $\text{Emb}_L$ and $\text{Emb}_A$, respectively. We use ResNet (He et al., 2016) for the visual encoder, $\text{Enc}_V$, to obtain visual features, $V_t \in \mathbb{R}^{w \times w \times d_v}$, and LSTM (Hochreiter and Schmidhuber, 1997) for the instruction encoder, $\text{Enc}_L$, to obtain instruction features, $\tilde{L} \in \mathbb{R}^{l \times d_l}$. We employ the bidirectional attention mechanism (Seo et al., 2017) for the cross attention Cross-Attn to align the visual and instruction features, and use the general attention Attn to align the action feature and each of fused visual and instruction features. See Appendix A.4 for the detailed descriptions of Cross-Attn and Attn modules.

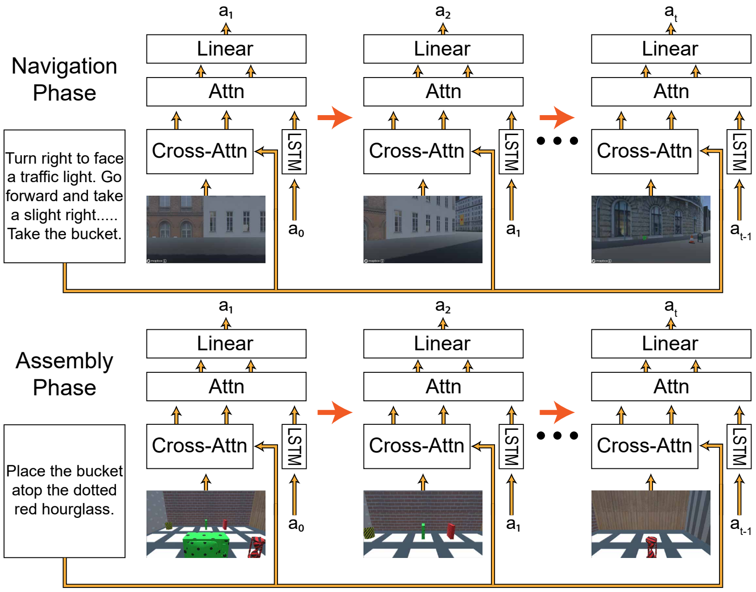

Navigation Phase
Assembly Phase
Turn right to face a traffic light. Go forward and take a slight right..... Take the bucket.
Place the bucket atop the dotted red hourglass.
Linear
Attn
Cross-Attn
LSTM
a₀
a₁
a₂
aₜ
aₜ₋₁

Figure 5.7: Vision-and-Language model: environment visual features, instruction language features, and action features are aligned to generate the next action.

We train the model with the teacher-forcing approach (Lamb et al., 2016) and cross entropy loss: $p_t(a_t) = \text{softmax}(\text{logit}_{a_t})$; $\mathcal{L} = -\sum_t \log p_t(a_t^*)$, where $a_t^*$ is ground truth action at time step $t$.

**Vision/Language only Baseline.** To check the uni-modality bias, we evaluate vision and language only baselines on our dataset. These exploit only single modality (visual or language) to predict the appropriate next action. To be specific, they use the same architecture as the Vision-and-Language baseline except the Cross-Attn module.

**Random Walk.** Agents take a random action at each time step without considering instruction and environment information.

**Shortest Path.** This baseline simulates an agent that follows the shortest path provided by A* algorithm (Hart et al., 1968) to show that the GT paths are optimal in terms of trajectory distances.

## 5.6 Experimental Setup

We split the dataset into train/val-seen/val-unseen/test-unseen. We assign the city sub-sections 1 to 5 to train and val-seen, sub-section 6 to val-unseen, and section 7 to test-unseen splits. We

| Model | Val Seen | | | | | | | Val Unseen | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Navigation | | | | | Assembly | | Navigation | | | | | Assembly | |
| | nDTW | CTC | | | | rPOD | PTC | nDTW | CTC | | | | rPOD | PTC |
| | | k=0 | k=3 | k=5 | k=7 | | | | k=0 | k=3 | k=5 | k=7 | | |
| V/L | 0.135 | 0.000 | 0.098 | 0.149 | 0.200 | 0.058 | 0.044 | 0.109 | 0.000 | 0.062 | 0.108 | 0.153 | 0.036 | 0.028 |
| V/O | 0.055 | 0.000 | 0.043 | 0.062 | 0.087 | 0.008 | 0.001 | 0.043 | 0.000 | 0.031 | 0.057 | 0.085 | 0.007 | 0.002 |
| L/O | 0.110 | 0.000 | 0.044 | 0.095 | 0.147 | 0.023 | 0.017 | 0.105 | 0.000 | 0.029 | 0.068 | 0.126 | 0.017 | 0.013 |
| R/W | 0.045 | 0.000 | 0.030 | 0.054 | 0.092 | 0.005 | 0.001 | 0.045 | 0.000 | 0.024 | 0.043 | 0.075 | 0.005 | 0.001 |
| S/P | 1.000 | - | 1.000 | 1.000 | 1.000 | - | - | 1.000 | - | 1.000 | 1.000 | 1.000 | - | - |
| H/W | 0.671 | 1.000 | 1.000 | 1.000 | 1.000 | 0.879 | 0.861 | 0.670 | 1.000 | 1.000 | 1.000 | 1.000 | 0.869 | 0.856 |

Table 5.3: Performance of baselines and humans on the metrics for the Val-Seen/Unseen splits. Overall, there is large human-model performance gap, indicating our ArraMon task is very challenging (V/L:Vision-and-Language, V/O:Vision-Only, L/O:Language-Only, R/W:Random-Walk, S/P:Shortest Path, H/W:Human-Workers).

| Model | Test Unseen | | | | | | |
|---|---|---|---|---|---|---|---|
| | Navigation | | | | | Assembly | |
| | nDTW | CTC | | | | rPOD | PTC |
| | | k=0 | k=3 | k=5 | k=7 | | |
| V/L | 0.114 | 0.000 | 0.082 | 0.122 | 0.168 | 0.047 | 0.035 |
| H/W | 0.664 | 1.000 | 1.000 | 1.000 | 1.000 | 0.884 | 0.873 |
| H/E | 0.806 | 1.000 | 1.000 | 1.000 | 1.000 | 0.992 | 0.990 |

Table 5.4: The Vision-and-Language (V/L) baseline and Human performance on Test-Unseen split (H/W:Human-Workers, H/E:Human-Expert).

randomly split data from sub-sections 1 to 5 into 80/20 ratio to get train and val-seen splits, respectively. Thus, the final number of task samples for each split is 4,267/1,065/1,155/1,205 (total: 17,068/4,260/4,620/4,820). The Stage 1 workers are equally distributed across the city sub-sections, so the dataset splits are not biased toward specific workers. We also keep the separate 2 sections (i.e., section 6 and 7) for the unseen dataset following Anderson et al. (2018b), which allows the evaluation of the models' ability to generalize in new environments. Note that for agents to proceed to the next phase, we allow them to pick up the closest target object (in the navigation phase) or place collected object at the closest location (in the assembly phase) when they do not perform the required actions. Training Details: We use 128 as hidden size. For word and action embedding sizes, we use 300 and 64, respectively. We use Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to 0.001 (see Appendix A.5.2 for details).

## 5.7 Results and Analysis

As shown in Table 5.3, overall, there is large human-model performance gap, indicating that our ARRAMON task is very challenging and there is much room for model improvement. Performance in the navigation and assembly phases are directly related. If perfect performance is assumed in the navigation phase, rPOD and PTC are higher than if there were low CTC-k scores in navigation (e.g., 0.382 vs. 0.044 for PTC of the Vision-and-Language model on val-seen: see Appendix A.6 for the comparison). This scoring behavior demonstrates that phases in our ARRAMON task are interweaved. Also, comparing scores from turn 1 and 2, all turn 2 scores are lower than their turn 1 counterparts (e.g., 0.222 vs. 0.049 nDTW of the Vision-and-Language model on val-seen split; see Appendix A.6 for the detailed turn-wise results). This shows that the performance of the previous turn strongly affects the next turn's result. Note that to relax the difficultly of the task, we consider CTC-3 (instead of CTC-0; see Section 5.2.2) as successfully picking up the target object and then we calculate the assembly metrics under this assumption. If this was not done, then almost all the metrics across assembly would be nearly zero.

### 5.7.1 Model Ablations

**Vision/Language Only Baseline.** As shown in Table 5.3, our Vision-and-Language baseline shows better performance over both vision-only and language-only models, implying our dataset is not biased to a single modality and requires multimodal understanding to get high scores.

**Random Walk.** The Random-Walk baseline shows poor performance on our task, implying that the task cannot be solved through random chance.

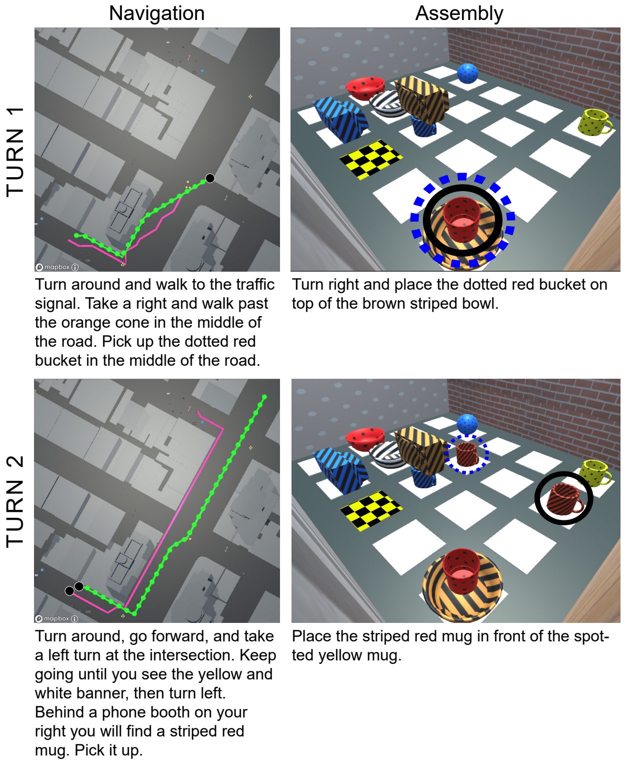
**Human Evaluation.** We conducted human evaluations with workers (Table 5.3, 5.4) as well as an expert (Table 5.4). For workers' evaluations, we averaged all the workers' scores for the verified dataset (from Stage 2: verification/following, see Sec. 5.3.1). For expert evaluation, we took 50 random samples from test-unseen and asked our simulator developer to blindly complete the task. Both workers and the expert show very high performance on our task (0.66 nDTW and 0.87 PTC for workers; 0.81 nDTW and 0.99 PTC for expert), demonstrating a large model-human performance gap and allowing much room for further improvements by the community on our challenging ARRAMON dataset.

### 5.7.2 Output Examples

As shown in an output example in Figure 5.8, our model navigates quite well and reaches very close to the target in the 1st turn and then places the target object in the right place in the assembly phase. However, in the 2nd turn, our model fails to find the "striped red mug" by missing the left turn around the "yellow and white banner". In the next assembling phase, the model cannot identify the exact location ("in front of the spotted yellow mug") to place the collected object (assuming the model picked up the correct object in the previous phase) possibly being distracted by another mug and misunderstanding the spatial relation. See Appendix A.7 for more output examples.

## 5.8 Conclusion

We introduced ARRAMON, a new joint navigation+assembling instruction following task in which agents collect target objects in a large realistic outdoor city environment and arrange them in a dynamic grid space from an egocentric view. We collected a challenging dataset (in English and now also Hindi) via a 3D synthetic simulator with diverse object referring expressions, environments, and visuospatial relationships. We also provided several baseline models which have a large performance gap compared to humans, implying substantial room for improvements by future work.



Turn around and walk to the traffic signal. Take a right and walk past the orange cone in the middle of the road. Pick up the dotted red bucket in the middle of the road.

Turn right and place the dotted red bucket on top of the brown striped bowl.

Turn around, go forward, and take a left turn at the intersection. Keep going until you see the yellow and white banner, then turn left. Behind a phone booth on your right you will find a striped red mug. Pick it up.

Place the striped red mug in front of the spotted yellow mug.

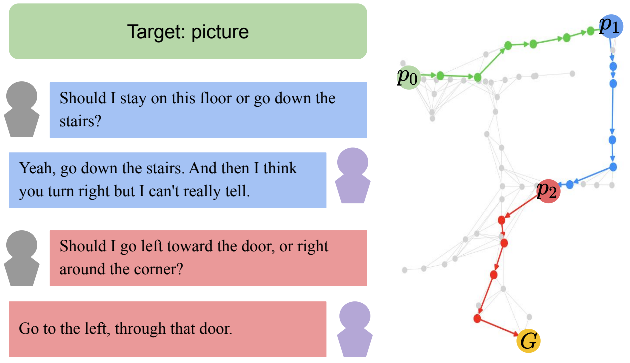
Figure 5.8: Visual demonstrations by our model in navigation and assembly phases (top-down view for illustration). GT navigation paths are solid pink lines and model's paths are dotted green lines (start = black dot). GT assembly target location is solid black circle and model's target object placement is dashed blue circle (start = checkered yellow tile, agent facing brick wall).

# CHAPTER 6: INSTRUCTION FOLLOWING EMBODIED NAVIGATION VIA FULL DIALOGUE

## 6.1 Introduction

With the increased number of intelligent agents being deployed in our daily lives, effective communication between humans and agents is becoming more important. Natural language is one of the most effective ways of communication due to its flexibility. Therefore, many efforts have been devoted to exploring the potential of its application in several tasks. Vision-and-Language Navigation (VLN) is one of the tasks in which agents have to navigate to a goal location in the indoor or outdoor environment by following natural language instructions (MacMahon et al., 2006; Tellex et al., 2011; Mei et al., 2016; Hermann et al., 2017; Brahmbhatt and Hays, 2017; Mirowski et al., 2018; Anderson et al., 2018b; Misra et al., 2018; Blukis et al., 2019; Thomason et al., 2019; Nguyen and Daumé III, 2019; Chen et al., 2019; Shridhar et al., 2020; Qi et al., 2020; Hermann et al., 2020; Berg et al., 2020; Ku et al., 2020).

While most VLN datasets only provide instructions from the oracle without considering the navigator's response, the useful Cooperative Vision-and-Dialogue Navigation (CVDN) (Thomason et al., 2019) dataset extends this one-way communication to two-way multi-turn dialogue (English) interaction between the oracle and the navigator. The dataset simulates a situation in which agents navigate through indoor environments towards a goal region by holding a conversation with humans for oracle guidance. Figure 6.1 shows an example in the CVDN dataset. Given the target information *"picture"* only, the navigator is asked to explore the environment by intuition (green path). The navigator can ask the oracle for assistance during navigation and then make progress (blue and red path) based on the oracle's response. From this dataset, Thomason et al. (2019) proposed the Navigation from Dialogue History (NDH) task, in which the agents are



Target: picture

Should I stay on this floor or go down the stairs?

Yeah, go down the stairs. And then I think you turn right but I can't really tell.

Should I go left toward the door, or right around the corner?

Go to the left, through that door.

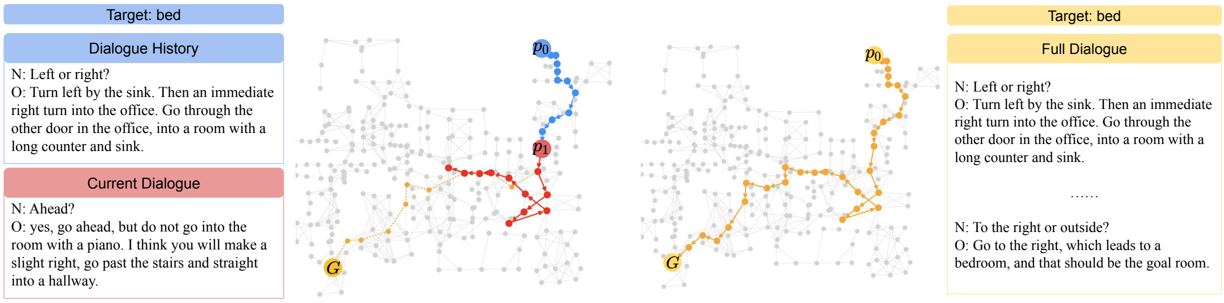
Figure 6.1: One example in the CVDN dataset. Given target information, dialogues in blue text and red text sequentially, the human navigates the green path, blue path, and red path accordingly.

asked to navigate toward the goal region $G$ given dialogue history and the current round of the dialogue. However, we find that this sub-path-based task setup does not provide enough supervision for the agent to reach the goal region $G$, and its primary evaluation metric – Goal Progress (GP) does not appropriately measure the agent's performance on the sub-path based task. In the example shown in Figure 6.1, one CVDN example is split into three navigation instances starting from $p_0, p_1, p_2$ and ends at $p_1, p_2, G$, respectively. One NDH instance only contains dialogue before the current navigation path (e.g., for navigation from $p_1$ to $p_2$, the agent only knows the target *"picture"* and the first round of the dialogue, which is in the blue box), thus lacks supervision for how to navigate from $p_2$ to the goal region $G$. However, the agent is evaluated with GP – the distance made towards the goal region $G$ from its starting point. This metric does not consider whether the agent follows the reference path. As a result, the agent could wander around to get a high GP score without following the path.

Hence, in this paper, we aim to redefine the NDH task via enhanced levels of supervision given to the agent, for better path fidelity while maintaining the advantage of learning from interactive dialogues. For this, we first build a strong state-of-the-art model based on Vision-Language transformers and pre-training, and illustrate that the current NDH task setup is not suitable for evaluating the agent's ability to follow natural language instructions. We show this by comparing the behaviors of the model on different evaluation metrics. Specifically, we find that a model with a higher GP score has a lower nDTW (normalized Dynamic Time Warping; Ilharco et al. (2019)) scores (see Table 6.3). Considering a high nDTW score reflects better path fidelity (and vice versa), pursuing high GP scores might not be suitable as an objective of an instruction-following navigation task. We attribute this mismatch to the aforementioned sub-path based task setup. Even though agents in the task could learn to navigate towards the target by commonsense and intuition, it might be hard to expect the agents to find the exact location of the target by using only their intuition (since this is hard even for human), especially in unseen environments since there is no specific regularity for target object placement (see Sec. 6.5.2 for analysis).

Therefore, we next propose a new task setup called NDH-FULL. We combine the sub-paths from the NDH task into the full path with the corresponding full dialogue, allowing the full supervision for agents on the instruction-following navigation task setup. As shown in the example of Figure 6.2, the NDH-FULL instance requires the agent to navigate from $p_0$ to $G$ with full dialogue instruction (i.e., target and multiple rounds of dialogues). In this setting, the agent has explicit supervision towards the goal region and is further faced with the challenge of understanding and grounding longer dialogues to navigate longer paths compared with the NDH task. We present a strong baseline model and several enhancement suggestions (based on curriculum learning, pre-training, and data-augmentation) for this task, and still leaves a large room for useful future work by the community on this challenging and realistic NDH-FULL task setup.

Our contributions are three-fold: (1) We first present a state-of-the-art model for the NDH task. (2) We then demonstrate that the NDH task setup lacks supervision for reaching the goal region and its primary evaluation metric does not capture the agent's path fidelity (via both qualitative and quantitative analysis). (3) Thus, we propose a new challenging and realistic task setup called NDH-FULL (along with strong baseline models), which provides full paths with the corresponding full dialogue; and enhances supervision to encourage path fidelity.



N: Left or right?
O: Turn left by the sink. Then an immediate right turn into the office. Go through the other door in the office, into a room with a long counter and sink.

N: Ahead?
O: yes, go ahead, but do not go into the room with a piano. I think you will make a slight right, go past the stairs and straight into a hallway.

N: Left or right?
O: Turn left by the sink. Then an immediate right turn into the office. Go through the other door in the office, into a room with a long counter and sink.

……

N: To the right or outside?
O: Go to the right, which leads to a bedroom, and that should be the goal room.

(a) NDH task setup

(b) NDH-FULL task setup

Figure 6.2: Comparison between the NDH task setup and the NDH-FULL task setup. Each sub-path corresponds to the sub-dialogue with same color. Dotted orange line in NDH task setup indicates shortest path between $p_1$ and goal region $G$. $N$ indicates Navigator and $O$ indicates Oracle in the dialogue.

## 6.2 Dataset Background and Task Setup

In this section, we discuss the vision-and-dialogue navigation task (NDH). We first introduce the CVDN dataset, and then show the two main issues of NDH and propose a new setup, NDH-FULL.

### 6.2.1 Cooperative Vision-Dialogue Navigation

The Cooperative Vision-and-Dialogue Navigation (CVDN) dataset contains dialogues between an oracle and a navigator. The navigator needs to find the target by asking questions during navigation. The oracle has access to the optimal navigation paths towards the target and responds to the navigator's questions. Specifically, each instance in the CVDN dataset contains a target object $t_0$, the start point for navigation $p_0$, the house scan $S$, the goal region $G$ where the target object is located in, multiple turns of utterances between the oracle and navigator, and the navigator's corresponding navigation trajectories after interacting with the oracle.

### 6.2.2 Navigation from Dialogue History (NDH)

**NDH Overview.** Based on the CVDN dataset, Thomason et al. (2019) defines the task of Navigation from Dialogue History (NDH). In the NDH task, the navigation path is the sub-path of the full navigation path in the CVDN dataset. As shown in Figure 6.2, the start point for this NDH instance is $p_1$. The dialogue before this start point is recorded as the dialogue history. The red path is what a human navigator traverses based on target information, dialogue history, the current round of the dialogue, and navigation history from $p_0$ to $p_1$. In NDH, the agent is asked to find the target located in the goal region $G$ based on this given information.

**Issues with NDH Task Setup.** Though many works (Hao et al., 2020; Zhu et al., 2020b; Wang et al., 2020; Zhang et al., 2020) have made great progress in finding the target, the NDH task setup still has a couple of issues. First, the NDH task asks the agent to find the target without providing enough supervision, which makes this task hard even for human to finish. One instance in NDH does not contain further dialogue turns. Thus, based on the information which is only limited to the oracle's response and no further following dialogue rounds, the navigator cannot reach the target even with human intuition about where the target might be in an unseen room environment. As shown in Figure 6.2, given target information, dialogue history, the current round of the dialogue, and navigation history, a human navigator can only traverse the red path, which is still far away from the goal region where the target locates.

Second, the NDH task uses Goal Progress (GP) as the main metric to evaluate the navigation agent, which does not encourage instruction following and is not appropriate for measuring the performance on sub-path based task. As shown in Figure 6.2, the shortest path between $p_1$ and $G$ does not align with the human's navigation according to dialogue information. The agent that navigates the shortest path or randomly explores the environment without following the instruction is not penalized by the GP metric. We show in Section 6.5.2 that the agent trained with the objective to have a higher GP will wander in the environment with long path length to get a GP without following the instruction, and thus deviates a lot from the reference path. This contradicts with the main goal of Vision-and-Language Navigation tasks which is to navigate environments by understanding instructions and grounding them with visual observations.

| Task | Split | # of Inst. | Avg. PL | Avg. DL |
|---|---|---|---|---|
| NDH | Train | 4742 | 7.68 | 3.82 |
| | Val-Seen | 382 | 7.61 | 4.31 |
| | Val-Unseen | 907 | 7.10 | 3.48 |
| | Test-Unseen | 1384 | - | 3.69 |
| | Total | 7415 | 7.59 | 3.78 |
| NDH-FULL | Train | 1145 | 25.82 | 5.79 |
| | Val-Unseen | 260 | 22.28 | 5.36 |
| | Test-Unseen | 248 | 24.42 | 5.56 |
| | Total | 1653 | 25.05 | 5.69 |

Table 6.1: Data statistics comparison between NDH and NDH-FULL. The average length of path and dialogue of NDH-FULL is longer than NDH's, implying NDH-FULL is a more challenging task. Since the path in Test-Unseen split of NDH task is not publicly released, the total Avg. PL is calculated except it (Inst.: instances, PL: path length, DL: dialouge length).

### 6.2.3 New Task Setup: NDH-FULL

In this section, we introduce the new task setup, NDH-FULL, to address the aforementioned issues in the NDH task. We create the NDH-FULL using the full dialogue-path pairs in CVDN. In other words, we combine multiple NDH instances that correspond to the same dialogue into one instance. As shown in Figure 6.2, given the target and full dialogue, the agent is asked to navigate from the start point $t_0$ to the goal region $G$. We also keep the sub-dialogue-path alignment information in the dataset, which brings the possibility for the agent to learn from sub-instructions. The NDH-FULL task setup provides full supervision for the agent to navigate towards the goal region and encourages the agent to understand long interactive dialogue and navigate with fidelity.

After combining all the sub-paths and dialogue turns into a full-length path-dialogue pair, the NDH-FULL has 1653 dialogue instances. We split them into training, validation-unseen, and test-unseen sets. We do not include validation seen set in NDH-FULL since we care more about agents' generalizability to unseen environments. The training, validation-unseen, and test-unseen sets contain 1145, 260, 248 instances respectively. Each of them is from 47, 10, and 10 non-overlapped scans, which preserves the important property that the environments of evaluation splits are unseen from the training set. We show detailed statistical comparison between NDH and NDH-FULL in Table 6.1. On average, the paths and dialogues of NDH-FULL are much

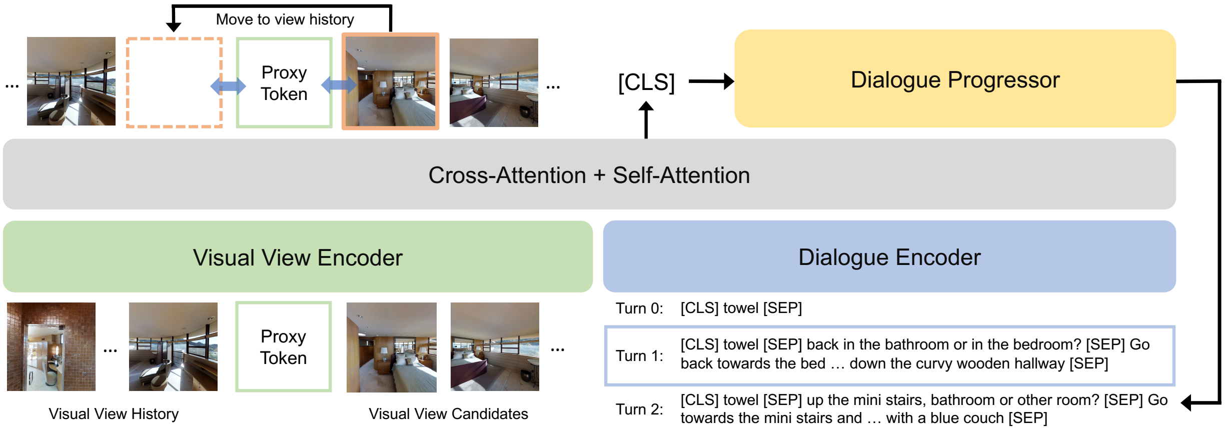

Move to view history
...
Proxy Token
...
[CLS]
Dialogue Progressor
Cross-Attention + Self-Attention
Visual View Encoder
Dialogue Encoder
...
Proxy Token
...
Visual View History
Visual View Candidates
Turn 0: [CLS] towel [SEP]
Turn 1: [CLS] towel [SEP] back in the bathroom or in the bedroom? [SEP] Go back towards the bed … down the curvy wooden hallway [SEP]
Turn 2: [CLS] towel [SEP] up the mini stairs, bathroom or other room? [SEP] Go towards the mini stairs and … with a blue couch [SEP]

Figure 6.3: The dialogue navigation model on NDH-FULL task. The next view to proceed is selected based on the attention score between the visual proxy token and the candidate views. The dialogue progressor takes the current and next dialogue round features and decides whether to move to the next round or stay.

longer than those of NDH (25.05 vs. 7.59 for path length, and 5.69 vs. 3.78 for dialogue length), which indicates that the NDH-FULL task setup is more challenging than NDH, allowing useful future work from the community. Furthermore, compared with NDH, the NDH-FULL gives the agent full supervision on how to reach the target and encourages the agent to understand long instructions and navigate based on the instructions.

## 6.3 NDH and NDH-FULL Models

We present the NDH task model and NDH-FULL task model in this section. To be specific, the NDH task model is built based on the vision-and-language transformer. Similar to the previous works (Hao et al., 2020; Hong et al., 2021), we employ LXMERT (Tan and Bansal, 2019) as the base architecture (Figure 6.3). The NDH-FULL task model takes the same architecture and is additionally equipped with the progressor module for moving through dialogue rounds. The NDH task model shows the state-of-the-art performance. However, by analyzing the behavior of the NDH task model on different metrics, we find the NDH task might not be suitable for evaluating the instructing-following navigation ability, thus, we propose the new NDH-FULL task and the baseline model (see Sec. 6.5.1 and 6.5.2).

**Pre-Training Model.** Pre-training is an effective approach to infuse prior knowledge in the vision-and-language navigation models (Majumdar et al., 2020; Hao et al., 2020; Hong et al., 2021). Compared with the previous works, our work proposes a new objective for pre-training. Instead of training the model with similarity score prediction (Majumdar et al., 2020) or discrete action label (Hao et al., 2020), we train the model with the objective that is nearly identical to the main navigation task for more effective transfer to the main task. Given a visual view sequence $V_t = \{v_1, v_2, ..., v_t\}$ and a corresponding navigation dialogue $D_i = \{d_{i0}, d_{i2}, ..., d_{i|D_i|}\}$, we train the model to select the next view to proceed among the candidates $C_t = \{c_{t1}, c_{t2}, ..., c_{t|C_t|}\}$. Additionally, we apply masked visual view prediction and masked language model loss as well. We employ ResNet (He et al., 2016) to get visual view features from panoramic images and use a multi-layer transformer to encode dialogue features like in LXMERT. The encoded features are fed to the LXMERT-based transformer module, TF$_{LXT}$.

$$L_n, L_v, L_l = \text{TF}_{LXT}([V_t^{\text{MASK}}; C_t], D_{1:i}^{\text{MASK}}) \tag{6.1}$$

where $L_n, L_v, L_l$ are the losses for navigation task, masked visual view prediction, and masked language model, respectively. $[;]$ is the concatenation operation, $V_t^{\text{MASK}} = \{v_1, v_{[\text{MASK}]}, ..., v_{[\text{MASK}]}, ..., v_t\}$ and $D_i^{\text{MASK}} = \{d_{i0}, d_{i1}, ..., [\text{MASK}], d_{i|D_i|}\}$ are masked visual view and dialogue features, respectively. $D_{1:i}$ is concatenation of the dialogue features up to the $i$th round. To compute the navigation loss, we use multi-head attention score (of the last layer) between the current visual view $v_t$ and the candidate visual views $C_t$ as the action logit following Hong et al. (2021). TF$_{LXT}$ consists of multiple layers of multi-head self-attention and cross-attention.

$$\hat{V}_t^j, \hat{D}_{1:i}^j = \text{MH-CrossATT}(V_t^j, D_{1:i}^j) \tag{6.2}$$

$$V_t^{j+1} = \text{MH-SelfATT}(\hat{V}_t^j) \tag{6.3}$$

$$D_{1:i}^{j+1} = \text{MH-SelfATT}(\hat{D}_{1:i}^j) \tag{6.4}$$

where MH-SelfATT is the multi-head self-attention and MH-CrossATT is the multi-head cross-attention. $V_t^j$ and $D_{1:i}^j$ are the input of visual view and dialogue features to the $j$th layer, respectively. The $l$th self/cross attention head at $j$th layer is computed by (for the visual view feature case):

$$a_{j,l} = \text{Softmax}(\frac{QK^\top}{\sqrt{d_h}})V \tag{6.5}$$

$$Q = W_{j,l}^q C_t^{j-1}, K = W_{j,l}^k V_t^{j-1},\ V = W_{j,l}^v V_t^{j-1} \tag{6.6}$$

$$V_t^j = [a_{j,1}; a_{j,2}; ...; a_{j,N_l}] \tag{6.7}$$

where $W_{j,l}^q$, $W_{j,l}^k$, and $W_{j,l}^v$ are trainable parameters, $d_h$ is hidden dimension, and $N_l$ is the number of attention heads. $C_t^{j-1}$ can be $V_t^{j-1}$ for self attention and $D_{1:i}^{j-1}$ for cross attention.

**NDH Model.** The dialogue navigation model for the NDH task shares the same base architecture as the pre-training model. On top of the pre-training model, we introduce the visual proxy token $p_t$ which links the candidate views to the current and past view history (i.e., the candidate views and the current/past view history only communicate with the proxy token via attention, but they do not directly interact with each other). It also plays as the recurrent state feature which maintains context history information. By introducing the visual proxy token, the view candidates' logits are calculated from the multi-head attention scores between the visual proxy token and the view candidates. The visual proxy token allows the model to consider both explicit (past view history) and implicit (recurrent state) context.

$$\hat{c}_t, \hat{p}_t = \text{TF}_{LXT}([V_t; p_t; C_t], D_{1:i}) \tag{6.8}$$

$$p_{t+1} = \text{Linear}(\hat{p}_t) \tag{6.9}$$

where $\hat{c}_t$ is the predicted view to proceed. The visual proxy token of the last output layer from the $\text{TF}_{LXT}$ model $\hat{p}_t$ is fed to a linear layer to become the visual proxy token at next time step.

**NDH-FULL Model.** For the NDH-FULL setup, we keep our strong NDH model as base architecture. In this model, we employ the CLIP visual feature (Radford et al., 2021) instead of the ResNet feature. To handle turns of the dialogue rounds, we introduce the dialogue progressor module which decides whether to move to the next round of the dialogue based on the current visual observation.

$$S_i = \text{Linear}([p_t; d_{i0}]) \tag{6.10}$$

$$S_{i+1} = \text{Linear}([p_t; d_{(i+1)0}]) \tag{6.11}$$

$$\text{NextTurn} = \begin{cases} i & \text{if } S_i \geq S_{i+1} \\ i+1 & \text{otherwise} \end{cases} \tag{6.12}$$

The dialogue progressor module simulates the situation in that the navigator is confused about which direction to go next and the oracle gives proper natural language guidance to the navigator. The progressor is trained from the alignment between sub-paths and corresponding dialogue rounds.

**Mixture of Imitation and Reinforcement Learning.** We use a mixture of imitation (IL) and reinforcement learning (RL) to train the model. For RL, we employ Actor-Critic (Mnih et al., 2016):

$$L_{IL} = -\sum_t \log p(a_t^*) \tag{6.13}$$

$$L_{RL} = -\sum_t (R_t - b_t) \log p(a_t^s) - \eta H(p(a_t)) \tag{6.14}$$

$$L_{MIX} = L_{IL} + \lambda L_{RL} \tag{6.15}$$

where $R_t$ is the discounted cumulative reward, $b_t$ is the baseline and $H(p(a_t))$ is the entropy term. $a_t^*$ is the teacher action and $a_t^s$ is the sampled action. We use distance-to-goal for the NHD task model and nDTW score for the NDH-FULL task model as the training rewards.

## 6.4 Experimental Setup

**Metrics.** We consider nDTW as the main metric of the new NDH-FULL task because nDTW reflects path fidelity better than other metrics (Ilharco et al., 2019). Other than nDTW, we also present evaluation results on success rate (SR), success weighted by path length (SPL), trajectory length (TL), and goal progress (GP) to allow evaluation from different perspectives.

**Training Details.** For the pre-training model, we use 9 language and 5 cross-modal LXMERT layers (but did not use their pre-trained weights), and use 768 as the hidden size. Following Tan and Bansal (2019), we use Adam (Kingma and Ba, 2015) as the optimizer with the learning rate $1 \times 10^{-4}$ and linear decay as in Devlin et al. (2019). We use L2 loss for visual view prediction, and cross-entropy loss for masked language model and next view selection. We use CVDN (Thomason et al., 2019), R2R (Anderson et al., 2018b), and a part of R2R's augmented data (Fried et al., 2018; Hao et al., 2020) as the training data. For the NDH task model, we use AdamW (Loshchilov and Hutter, 2018) as the optimizer with the learning rate $1 \times 10^{-5}$. Only CVDN data is used for fine-tuning the model. In the NDH-FULL task, we do not apply pre-training for the full-dialogue model. We use ResNet-152 feature and ResNet50-based CLIP feature. All the experiments are run using the NVIDIA TITAN Xp / GeForce GTX 1080 Ti / GeForce RTX 2080 Ti GPUs. We use PyTorch (Paszke et al., 2017) to build all models. We use manual tuning (e.g, learning rate=$\{1 \times 10^{-3}$, ..., $1 \times 10^{-6}\}$, and the layers of the transformer model=$\{$5(cross-modal)/3(language), 9/5$\}$) for selecting hyper-parameters. The number of trainable parameters of our NDH and NDH-FULL task models are 181M and 182M, respectively.

## 6.5 Results

### 6.5.1 State-of-the-Art Results on NDH Task

In this section, we present our model's performance on the NDH task. As shown in Table 6.2, our model outperforms all the state-of-the-art models on the primary evaluation metric – Goal

| Models | Val Unseen | Test Unseen |
|---|---|---|
| PREVALENT (Hao et al., 2020) | 3.15 | 2.44 |
| CMN (Zhu et al., 2020b) | 2.97 | 3.14 |
| EAML (Wang et al., 2020) | 4.65 | 3.91 |
| BabyWalk (Zhu et al., 2020a) | - | 4.46 |
| Ours | **5.51** | **5.27** |

Table 6.2: Performance on NDH task measured with Goal Progress. Our model outperforms all the state-of-the-art models in the validation unseen environment and ranks 1st (at the time of EMNLP 2021 submission deadline) on the NDH task leaderboard (‘s-agent’ team). EAML: Environment-Agnostic Multitask Learning.

Progress by a large margin and ranks 1st (at the time of EMNLP 2021 submission deadline) on the leaderboard (‘s-agent’ team).[1] This shows that our model performs strongly on the navigation task.

### 6.5.2 Analyzing the Issue in NDH Task Setup

However, we believe that the NDH task is not evaluated appropriately via the primary metric (i.e., GP) since GP could not reflect the instruction-following ability of the agents in the task. We conduct an experiment by running our model with two different rewards for reinforcement learning: global target reward and local target reward. In global target reward, the agent gets a positive reward if it moves closer to the final target region, and a negative reward otherwise. In local target reward, the agent receives the reward based on whether it moves closer to the final position of the sub-path. Since there is no explicit instruction for the path between the final position of the sub-path and the global target region (except when the sub-dialogue-path pair is the last pair in the full dialogue), the global target model stands for a model trained with implicit navigation supervision towards the global target region and the local target model stands for a model trained with no such implicit navigation supervision towards the global target region. We show the results in Table 6.3.

[1] https://eval.ai/web/challenges/challenge-page/463/leaderboard/1292

| Reward Type | GP | SR | TL | nDTW | nDTW+ |
|---|---|---|---|---|---|
| Global Target | 5.51 | 19.8 | 24.582 | 0.253 | 0.243 |
| Local Target | 3.82 | 37.2 | 10.591 | 0.518 | 0.287 |

Table 6.3: Performance on NDH task (val-unseen split). Global Target is the model with a reward of distance to the final target and Local Target is the one with a reward of distance to the end point of the sup-path of each data instance in NDH task (nDTW+: nDTW based on extended reference path up to the target location).

**Goal Progress.** The GP score of the global target model is much higher than the local target model (5.51 vs. 3.82), indicating that the global target model reaches closer to the global target location with implicit supervision.

**Instruction Following.** However, when we compare the success rate scores (19.8 vs. 37.2) and nDTW scores (0.253 vs. 0.518), the local target model outperforms the global target model, indicating that the local target model follows the reference path better. This mismatch in metrics implies that GP cannot measure the agent's ability to follow the path well.

**Intuition to Reach Target.** A higher GP score of the global target model can be considered as the result of learning intuition to navigate towards the target region without explicit supervision. However, we show in Table 6.3 that the global target model has a much higher trajectory length (TL) compared with the local target model (24.582 vs. 10.591), indicating that the agent learns to get a higher GP by wandering in the environment rather than proceeding towards a specific direction with intuition. We also show that the global target model has a lower nDTW+ score (which is a nDTW score against the extended reference path to the target location measuring the agent's ability to follow the path from the current starting point to the target) compared with the local target model (0.243 vs. 0.287), which also supports the observation that the global target model does not follow the extended path towards the global target region with intuition to get a high GP score.

Therefore, pursuing higher GP scores might not reflect agents' ability to interpret and follow given dialogues. For this reason, we introduce a new task setup, NDH-FULL, which encourages instruction following by giving full supervision towards the global target to the agent.

| Models | Val-Unseen | | | | |
|---|---|---|---|---|---|
| | GP | SR | SPL | TL | nDTW |
| Random-Walk | 5.755 | 3.1 | 2.8 | 10.056 | 0.141 |
| No-Dialogue | 10.972 | 6.5 | 5.6 | 29.556 | 0.267 |
| Target-Only | 10.005 | 6.2 | 4.9 | 29.828 | 0.267 |
| Full-Dialogue | 11.124 | 7.7 | 6.2 | 32.678 | 0.277 |
| | Test-Unseen | | | | |
| | 14.045 | 10.5 | 7.6 | 28.539 | 0.301 |

Table 6.4: Performance on the new NDH-Full task. The models are selected according to the best nDTW scores.

| Models | Val-Unseen | | | | |
|---|---|---|---|---|---|
| | GP | SR | SPL | TL | nDTW |
| Curriculum Learning | 11.241 | 7.3 | 6.3 | 32.169 | 0.273 |
| Pre-Training | 12.268 | 7.3 | 6.2 | 34.166 | 0.278 |
| Data Augmentation | 11.058 | 6.9 | 5.9 | 35.306 | 0.263 |

Table 6.5: Performance from different approaches on the new NDH-Full task.

### 6.5.3 NDH-Full Task Results & Suggestions

We show the performance of our model and its ablations on the new NDH-Full task. We experiment with the "Random-Walk" baseline which chooses a random heading and walks up to 5 steps forward as in Thomason et al. (2019), "No-Dialogue" baseline which only considers visual input, and "Target-Only" baseline which considers visual input and the target information. As shown in Table 6.4, with full supervision towards the target goal region (Full-Dialogue), the agent outperforms the other baselines in all metrics, which indicates that full-dialogue provides useful supervision for the agent. However, performance gap between models is not large. Considering the full-dialogue model shows the best performance in the NDH task, the new NDH-Full task is quite challenging with longer paths and dialogues. Moreover, requirement of aligning each sub-path and the corresponding dialogue round in the NDH-Full task introduces additional dimension of difficulty to handle for better performance in instruction-following navigation.

Therefore, we believe there is still a large room for potential improvement by applying more advanced approaches. Thus, we experiment with some of the advanced approaches here as an initial step to tackle this challenge.

**Curriculum Learning.** We divide one data instance into multiple instances so that each resulting data point has a different number of dialogue rounds and a corresponding sub-path (i.e., 2, 3, and 4 or more than 4 dialogue rounds) and train the model on the subset of the data and move on to the longer dialogue/path ones (starting from the 2 dialogue rounds to the original full dialogue rounds). But, as shown in Table 6.5, this curriculum learning approach only does not show an improvement. With a more finely designed learning procedure, we believe curriculum learning would help improve the performance on the challenging new task.

**Pre-Training.** We also apply the pre-trained weights which are used for the NDH model. However, this also does not give any distinct performance boost. This might be because the pre-training model for the NDH task is passive in that the model is given visual and textual features at once. On the other hand, in the NDH-FULL task, agents should actively ask for guidance when they are confused. Therefore, aligning dialogue rounds with the visual observation from the environment is one challenging factor in the new task.

**Data Augmentation.** The data size of NDH-FULL shrinks after combining all sub-paths and dialogue rounds (7415 vs. 1653, see Table 6.1). To compensate for the loss, we try data augmentation by generating the oracle's instruction with the speaker model (Fried et al., 2018; Tan et al., 2019b). We modify their speaker model to take the context (i.e., dialogue history) as well as view trajectory to fit to the CVDN dataset. We replace the oracle's instruction in a round of dialogue with the newly generated ones to give the model more diverse forms of instructions. But, we do not see an improvement from training the model on this augmented data possibly because NDH-FULL requires accurate instructions to navigate quite long paths and the quality of the current speaker model could not meet the criteria. This allows future work on more effective generation methods.

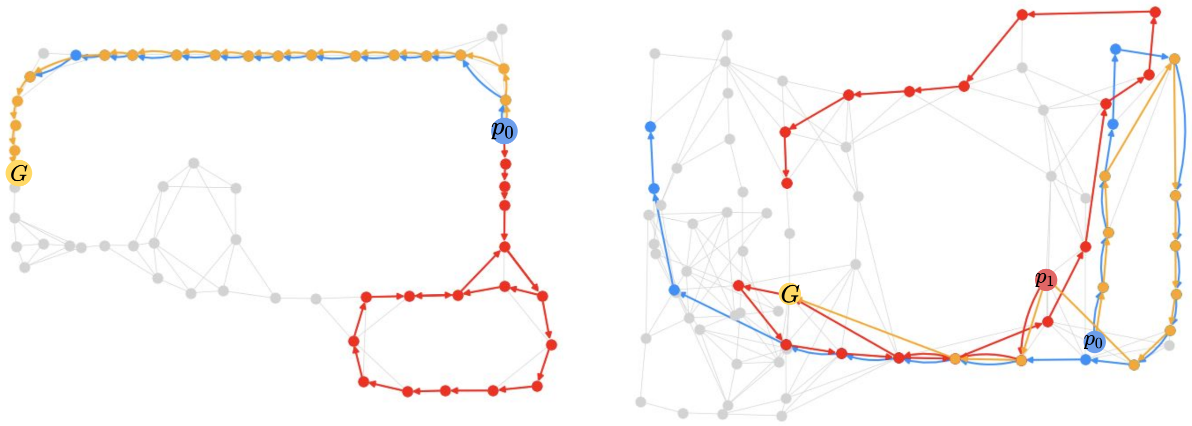


Figure 6.4: Trajectory comparison between the NDH task model (red line) and the NDH-FULL task model (blue line). Yellow line is the reference path ($p_0$: starting point of the whole path, $p_1$: starting point of 2nd sub-path, $G$: goal point).

### 6.5.4 Trajectory Comparison

As shown from the top figure in Figure 6.4, the NDH task agent (red line) fails to follow the correct reference trajectory (yellow line) by misunderstanding the oracle's instruction (*"turn around and follow the red carpet path. Once you pass a vase on your left stop"*) while still getting a positive GP score (8.820). On the other hand, the NDH-FULL task agent (blue line) can manage to follow the instructions showing a high path fidelity (nDTW score: 0.735). This example implies that GP is not a good metric for measuring instruction-following. In the bottom example, the NDH task agent starts from $p_1$ (in the sub-path task setup) and move towards the goal location, but it directly passes the target object and wanders in the room. This trajectory deviates much from the reference sub-path, but the agent still gets a high GP (8.226) since it finally stops near the goal region. Though the NDH-FULL task agent doesn't stop at the goal region either, it follows the reference path well during most of the navigation process (nDTW score: 0.549).

## 6.6 Conclusion

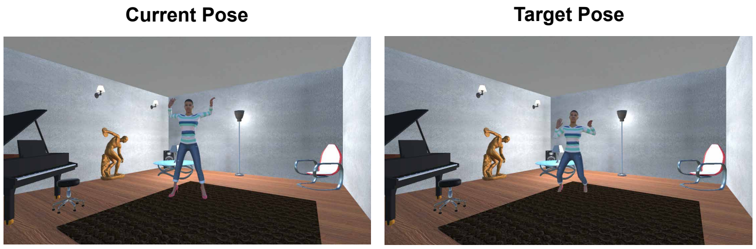
We explored the NDH task, which is built on the useful Cooperative Vision-and-Dialogue Navigation (CVDN) dataset, and found the mismatch between the task setup and evaluation by analyzing the scoring behaviors of our state-of-the-art model. Therefore, we proposed a new task called NDH-FULL. We combined all split paths and dialogue rounds of NDH to create the full path and dialogue, resulting NDH-FULL has longer paths and dialogues than NDH and it makes NDH-FULL more challenging. We also presented a baseline model, resulting scores, and suggestions for promising research directions on the NDH-FULL task.

# CHAPTER 7: POSE CORRECTIONAL CAPTIONING AND RETRIEVAL

## 7.1 Introduction

As the well-being trend grows and people globally move to a new online lifestyle, interest in remotely (i.e., at home or in the office) learning health and exercise activities such as yoga, dance, and physical therapy is growing. Through advanced video streaming platforms, people can watch and follow the physical movements of experts, even without the expert being physically present (and hence scalable and less expensive). For such remote activities to be more effective, appropriate feedback systems are needed. For example, a feedback system should catch errors from the user's movements and give proper guidance to correct their poses. Related to this line of work, many efforts have been made on human pose estimation and action recognition (Johnson and Everingham, 2010, 2011; Andriluka et al., 2014; Toshev and Szegedy, 2014; Wei et al., 2016; Andriluka et al., 2018; Yan et al., 2018b; Zhao et al., 2019; Cao et al., 2019; Sun et al., 2019; Verma et al., 2020; Rong et al., 2020). Research on describing the difference between multiple images has also been recently active (Jhamtani and Berg-Kirkpatrick, 2018; Tan et al., 2019a; Park et al., 2019; Forbes et al., 2019). However, there has been less focus on the human pose-difference captioning tasks, which require solving unique challenges such as understanding spatial relationships between multiple body parts and their movements. Moreover, the reverse task of retrieving or generating a target pose is also less studied. Combining these two directions together can allow for more interweaving human-machine communication in future automated exercise programs.

Relatedly, interest in embodied systems for effective human-agent communication is increasing (Kim et al., 2018; Wang et al., 2019a; Abbasi et al., 2019; Kim et al., 2020c). Embodiment is also a desirable property when designing virtual assistants that provide feedback. For example, embodied virtual agents can show example movements to users or point at the users' body parts



**Description 1 (English):** slide your right foot back one step and bend your knees, bring your wrists closer to your shoulders but maintain the position of your hands, finally drop your arms at the shoulder to level your hands with your neck.
**Description 2 (English):** bend both of your legs. bring both of your arms down almost below your ears. your left palm should be facing towards the chair. the back of your right hand should be facing the glass table.
**Description 3 (English):** bend both knees away from the lamp, lower down your body towards the rug, bring both hands down above your shoulder, right palm facing front and left palm facing the chair, tilt your head back a little towards the lamp.
**Description 1 (Hindi):** अपने दाहिने पैर को एक कदम पीछे खिसकाएं और अपने घुटनों को मोड़ें, अपनी कलाई को अपने कंधों के करीब लाएं लेकिन अपने हाथों की स्थिति को बनाए रखें, अंत में अपनी गर्दन के साथ अपने हाथों को समतल करने के लिए अपने हाथों को कंधे पर रखें।

Figure 7.1: Current and target image pair and the corresponding correctional descriptions in both English and Hindi (we show only one of the three Hindi descriptions due to space).

that need to move. Furthermore, for effective two-way communication with embodied agents, reverse information flow (i.e., human to agents) is also needed. A user may want to describe what actions they took so that the agent can confirm whether the user moved correctly or needs to change their movement. The agent should also be able to move its body to match the pose that the user is describing to help itself understand.

Therefore, to encourage the multimodal AI research community to explore these two tasks, we introduce a new dataset on detailed pose correctional descriptions called FIXMYPOSE (फिक्समाइपोज़), which consists of image pairs (a "current" and "target" image) and corresponding correctional descriptions in both English and Hindi (Fig. 7.1). To understand our dataset, imagine you are in a physical therapy program following an instructor in a prerecorded video at home. Your movements and resulting pose are likely to be wrong, hence, you would like a virtual AI assistant to provide detailed verbal guidance on how you can adjust to match the pose of the instructor. In this case, your incorrect pose is in the "current" image and the pose of the instructor is in the "target" image, forming a pair. The verbal guidance from the virtual AI assistant is the correctional description.

From our FIXMYPOSE dataset, we introduce two tasks for multimodal AI/NLP models: the ‘pose-correction-captioning’ task and the ‘target-pose-retrieval’ task. In the pose-correction-captioning task, models are given the “current” and “target” images and should generate a correctional description. The target-pose-retrieval task is the reverse of the pose-correction-captioning task, where models should select the correct “target” image among other distractor images, given the “current” image and description. This two-task setup will test AI capabilities for both important directions in pose correction (i.e., agents generating verbal guidance for human pose correction, and reversely predicting/generating poses given instructions), to enable two-way communication between humans and embodied agents in future research. To generate image pairs, we implement realistic 3D interior environments (see Sec. 7.3 for details). We also extract body joint data from characters to allow diverse tasks such as pose-generation (Fig. 7.5). We collect descriptions for these image pairs by asking annotators from a crowdsourcing platform to explain to the characters how to adjust their pose shown in the “current” image to the one shown in the “target” image in an instructional manner from the characters’ egocentric view (see Table 7.1). Furthermore, we ask them to refer to objects in the environment to create more detailed and accurate correctional descriptions, adding diversity and requiring models to understand the spatial relationships between body parts and environmental objects. The descriptions also often describe movement indirectly through implicit movement descriptions and analogous references (e.g., ”like you are holding a cane”) (see Sec. 7.4.2), which means AI models performing this task should develop a commonsense understanding of these movements and references. To encourage multimodal AI systems to expand beyond English, we include Hindi descriptions as well (Fig. 7.1).

Empirically, we present both unimodal and multimodal baseline models as strong starting points for each task, where we apply multiple cross-attention layers to integrate vision, body-joints, and language features. For the pose-correction-captioning model, we employ reinforcement learning (RL), which uses self-critical sequence training (Rennie et al., 2017), for further

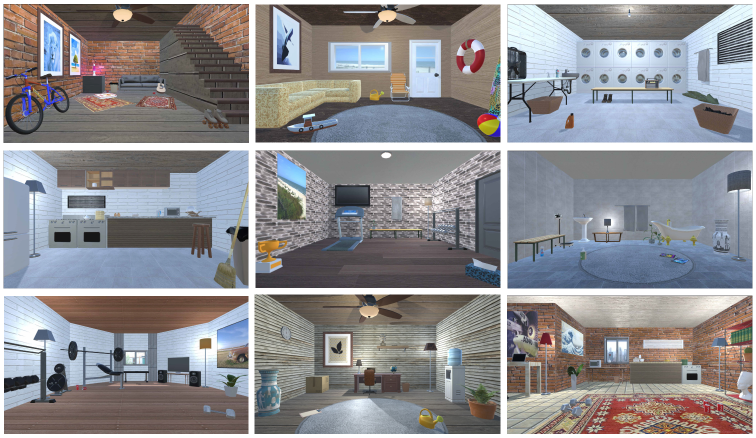
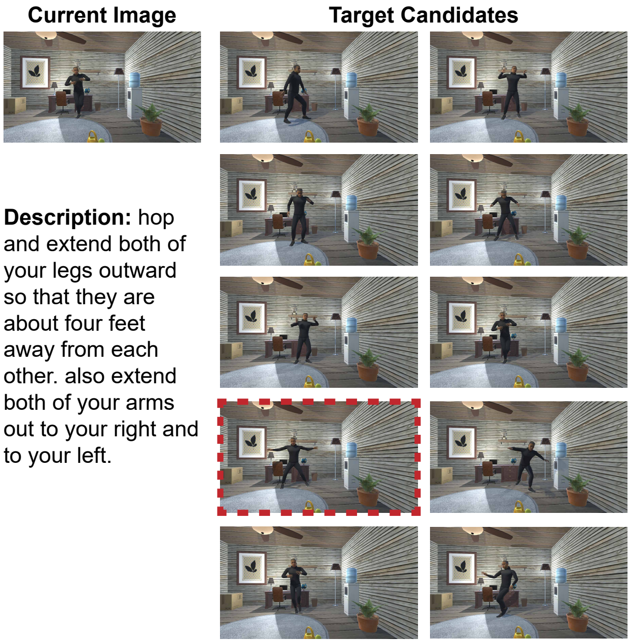


Figure 7.2: Example room environments: each room has a diverse style/theme (e.g., office, bathroom, living room).

improvement. Also, we present the results from a multilingual training setup (English+Hindi) which uses fewer parameters by sharing model components, but shows comparable scores.

The multimodal models in both tasks show better performance than unimodal models, across both qualitative human evaluation and several of the evaluation metrics, including our new task-specific metrics: object, body-part, and direction match (details in Sec. 7.7.1). There is also a large human-model performance gap on the tasks, allowing useful future work on our challenging dataset. We also show balanced scores on demographic ablations, implying that our dataset is not biased toward a specific subset. Furthermore, our model performs competitively with existing works when evaluated on other image-difference datasets (Image Editing Request (Tan et al., 2019a), NLVR2 (Suhr et al., 2019), and CLEVR-Change (Park et al., 2019)). Finally, to verify the simulator-to-real transfer of our FIXMYPOSE dataset, we collect a test-real split which consists of real-world image pairs and corresponding descriptions, and show promising performance on the real images.

Our contributions are 3-fold: (1) We introduce a new dataset, FIXMYPOSE, to encourage research on the integrated field of human pose, correctional feedback systems on feature differences with spatial relation understanding, and embodied multimodal virtual agents; (2) We collect a multilingual (English/Hindi) dataset; (3) We propose two tasks based on our FIXMY-



**Description:** hop and extend both of your legs outward so that they are about four feet away from each other. also extend both of your arms out to your right and to your left.

Figure 7.3: The target-pose-retrieval task: models have to select the correct “target” image from a set of distractors (the image with red dashed border is the ground-truth target pose), given “current” image and correctional description.

POSE dataset (pose-correction-captioning and target-pose-retrieval), and present several strong baselines as useful starting points for future work (and also demonstrate sim-to-real transfer).

## 7.2 Tasks

**Pose Correctional-Captioning Task.** During this task, the goal is to generate natural language (NL) correctional descriptions, considering the characters’ egocentric view, that describe to a character how they should adjust their pose shown in the “current” image to match the pose shown in the “target” image (Fig. 7.1). As the “current” and “target” image pairs contain various objects in realistic room environments, models should have the ability to understand the spatial relationships between the body parts of characters and the environment from the characters’ perspectives.

**Target Pose Retrieval Task.** Here, the goal is to select the correct “target” image among 9 incorrect distractors, given the “current” image and the corresponding correctional description (Fig. 7.3). For the distractor images, we only consider images that are close to the “target”

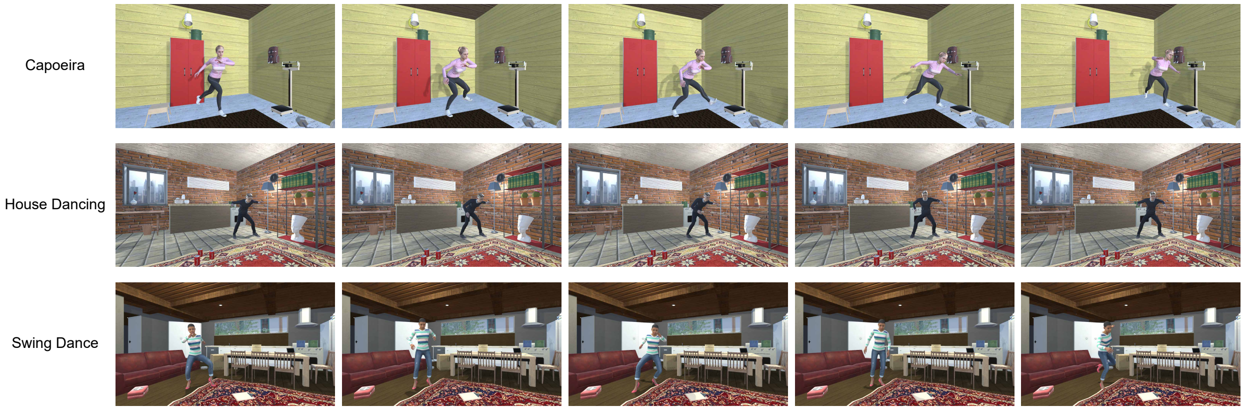


Figure 7.4: Examples of specific movement animations (each image is 10 frames apart). Each image sequence show a segment of the movement animation.

pose in terms of body joints distances (see Appendix A.8 for detailed criteria). These distractor choices discourage models from easily discerning the correct "target" image via shallow inference or shortcuts, requiring minute differences to be captured by models. The large human-model performance gap (Sec. 7.7.2) verifies the quality of our distractors.

## 7.3 Dataset

Our FixMyPose dataset is composed of image pairs with corresponding correctional descriptions in English/Hindi.

**Image, 3D Body Joints, and Environment Generation.** We create 25 realistic 3D diverse room environments, filled with varying items (Fig. 7.2). To ensure diversity, we employ 6 human character avatars of different demographics across gender/race (each character is equally balanced in our dataset).[1] Since creating/modifying the body of characters requires 3D modeling/artistic expertise, we use pre-made character models that are publicly available (hence also copyright-free for our community's future use) in Adobe's Mixamo[2]. In the rooms, the characters perform

[1] Our task focuses on understanding body movements/angles and not demographics, but we still ensure demographic diversity and balance in our dataset for ethical/fairness purposes so as to avoid unintended biases (e.g., see the balanced demographics ablation results and Sim-to-Real Transfer results on people with different demographics with respect to the 6 character avatars in Sec. 7.7). We plan to further expand our dataset with other types of diversity (e.g., height, age) based on digital avatar availability.

[2] https://www.mixamo.com

20 movement animations and the camera captures images on a fixed interval (Fig. 7.4). We also obtain 3D positional body joint data of the character's poses in the "current" and "target" images to provide additional useful features and allow a potential reverse pose-generation task (Fig. 7.5). See Appendix A.9.1 for more on environment creation, body joint data, and image capturing.

**Description Collection.** We employ annotators from the crowdsourcing platform Amazon Mechanical Turk[3] to collect the correctional descriptions. Workers are provided 3 images, "current", "target" images, and a "difference" image that shows the difference between the two images, allowing them to write clear descriptions (see Appendix A.9 for the images and collection interface). We ask them to write as if they are speaking to the characters as assistants who are helping them (like "You should ..."), not calling them by the 3rd person (like "The person ...", "They/She/He ..."). It also helps prevent accidental biased terms assuming the demographics of the characters. We collect 1 description for each image pair for the train split and 3 for all subsequent splits (i.e., val-seen/val-unseen/test-unseen) from unique workers, making the computation of automated evaluation metrics such as BLEU possible.

**Description Verification.** Each description and its corresponding image pair is given to a separate group of workers through a verification task. For each description, 3 different workers are asked to rank it from 1-4 based on its relevance to the image pair and its clarity, similar to previous works (Lei et al., 2020a). Descriptions that 2/3 of the workers rate lower than 3 are discarded. Image pairs that are flagged with certain issues are discarded as they do not provide good data (see Appendix A.9.2 for the verification interface and flags).

**Hindi Data Collection.** To collect the translated Hindi descriptions, we present a translation task to workers. Workers are given a description that has passed the verification task and its corresponding image pair to ensure the original meaning is not lost (see Appendix A.9.2 for the translation interface).

**Worker Qualification and Payment.** We require workers completing either of the tasks to be fluent in the needed languages and to have basic MTurk qualifications. The writing task takes

[3] https://www.mturk.com

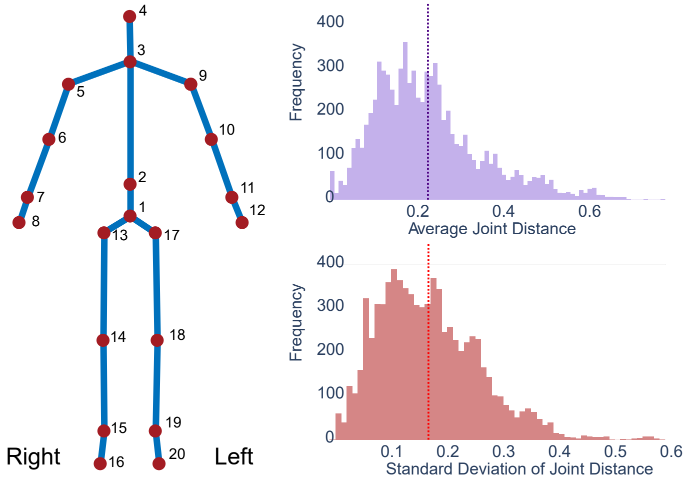


Figure 7.5: The 3D joint configuration of characters (left). The distribution of joint distances (meters) between poses of the "current" and "target" images (right). The Avg. of Min joint distances: 0.04 and the Avg. of Max joint distances: 0.65.

around 1 minute and workers are paid $0.18 per description. To encourage workers to write more and better descriptions, an additional increasing-bonus system is implemented. See Appendix A.9.4 for qualification/bonus/payment details.

## 7.4 Data Analysis

We collect 7,691 image pairs and 11,127 correctional descriptions for both English and Hindi (1 per train and 3 per evaluation splits). Our dataset size is comparable to other captioning tasks/-datasets such as Image Editing Request (Tan et al., 2019a) (3.9K image pairs/5.7K instructions), Spot-the-Diff (Jhamtani and Berg-Kirkpatrick, 2018) (13.2K image pairs/captions), and Birds-to-Words (Forbes et al., 2019) (3.3K image pairs/16K paragraphs). We plan to keep extending the dataset and add other languages in the future.

### 7.4.1 Statistics

**Joint Distances.** Fig. 7.5 shows the distribution of average joint distances (meters) between the poses in the "current" and "target" images. As indicated by the mean (0.24m), stddev (0.18m),

| Reference Frame | Freq. | Example (English) |
|---|---|---|
| Egocentric Relation | 100% | "... **rotate your left shoulder** so that your hand is **above your elbow** ..." |
| Environmental Direction | 52% | "... turn your left leg and right leg to the left to **face the wall with the door** ..." |
| Implicit Movement Description | 58% | "... lean your body towards and slightly over your right leg ..." |
| Analogous Reference | 18% | "... in front of you **as if you are gesturing for someone to stop** ..." |

Table 7.1: Examples of linguistic properties in correctional descriptions (see Appendix A.10.3 for examples and image examples of implicit movement description).

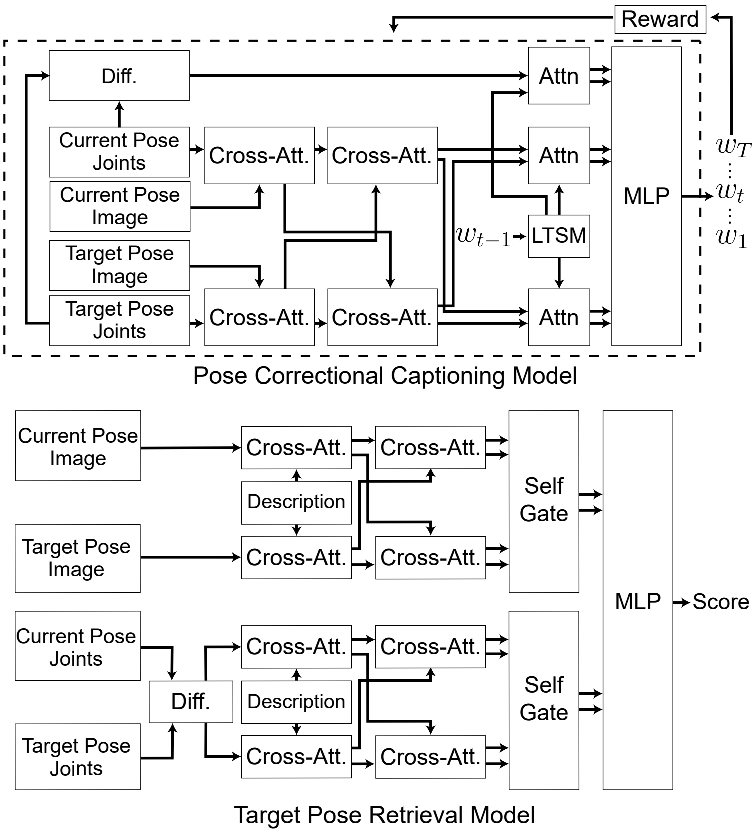
and min/max (0.04/0.65m) of the average distance of individual joints, models should be able to capture different movement levels simultaneously in an image pair.

**Description Vocabulary and Length.** The collection of descriptions in our FixMyPose dataset has 4,045/4,674 unique English/Hindi words. The most common words in both languages (see Appendix A.10.1 for details and pie charts) relate to direction, body parts, and movement, showing that models need to have a sense of direction with respect to body parts and objects, and also capture the differences between the poses to infer the proper movements. The average length of the multi-sentenced descriptions (49.25/52.74 words) is high, indicating that they are well detailed (see Appendix A.10.2 for details).

### 7.4.2 Linguistic Properties

To investigate the diverse linguistic properties in our dataset, we randomly sample 50 descriptions and manually count occurrences of traits. We found interesting traits (see Table 7.1 and Appendix A.10.3 for examples), requiring agents to deeply understand characters' movements and express them in an applicable form (the Hindi descriptions also share these traits).

**Egocentric and Environmental Direction.** Descriptions in our FixMyPose dataset are written considering the egocentric (first-person) view of the character. Descriptions also reference many environmental objects and their relation to the characters' body parts, again from an egocentric



Pose Correctional Captioning Model

Target Pose Retrieval Model

Figure 7.6: The pose-correction-captioning model (top) and the target-pose-retrieval model (bottom).

view. This means models must understand spatial relations of body parts and environmental features from the egocentric view of the character rather than the view of the "camera".

**Implicit Movement Description and Analogous Reference.** Implicit movement description and analogous reference are often present in descriptions. These descriptions imply movements without needing to say them. Analogous references are a more extreme form of implicit movement description, where the movement is wrapped in an analogy. Models must develop commonsense knowledge of these movements in order to understand their meaning. See Table 7.1 and Appendix A.10.3 for examples.

## 7.5 Models

We present multiple strong baselines for both the pose-correction-captioning and target-pose-retrieval task (Fig. 7.6) to serve as starting points for future work.

### 7.5.1 Pose Correctional Captioning Model

We employ an encoder-decoder model for the pose-correction-captioning task. Also, we apply reinforcement learning (RL) after training the encoder-decoder model, and present multilingual training setup which reduces the number of parameters through parameter sharing.

**Encoder.** We employ ResNet (He et al., 2016) to obtain visual features from images. To be specific, we extract feature maps $f^c$ and $f^t \in \mathbb{R}^{N \times N \times 2048}$ from the "current" pose image $I^c$ and the "target" pose image $I^t$, respectively: $f^c = \text{ResNet}(I^c)$; $f^t = \text{ResNet}(I^t)$. For 3D joints, $J^c, J^t \in \mathbb{R}^{20 \times 3}$, we use linear layer to encode: $\hat{J}^c = \text{PE}(W_j^\top J^c)$; $\hat{J}^t = \text{PE}(W_j^\top J^t)$; $J^d = \text{PE}(W_j^\top (J^t - J^c))$, where $W_j$ is the trainable parameter (all $W_*$ from this point on denote trainable parameters) and PE (Gehring et al., 2017; Vaswani et al., 2017) denotes positional encoding.

**Decoder.** Words from a description, $\{w_t\}_{t=1}^{T}$, are embedded in the embedding layer: $\hat{w}_{t-1} = \text{Embed}(w_{t-1})$, then sequentially fed to the LSTM layer (Hochreiter and Schmidhuber, 1997): $h_t = \text{LSTM}(\hat{w}_{t-1}, h_{t-1})$. We employ the bidirectional attention mechanism (Seo et al., 2017) to align image features and joints features.

$$\tilde{f}^c, \tilde{J}^t, \tilde{f}^t, \tilde{J}^c = \text{CA-Stack}(f^c, \hat{J}^c, f^t, \hat{J}^t) \tag{7.1}$$

where CA-Stack is a cross attention stack (see Appendix A.11).

$$f = W_c^\top[\tilde{f}^c; \tilde{f}^t; \tilde{f}^c \odot \tilde{f}^t],\ J = W_c^\top[\tilde{J}^c; \tilde{J}^t; \tilde{J}^c \odot \tilde{J}^t] \tag{7.2}$$

$$f_t = \text{Att}(h_t, f),\ J_t = \text{Att}(h_t, J),\ J_t^d = \text{Att}(h_t, J^d) \tag{7.3}$$

$$k_t = W_k^\top[f_t; J_t; h_t; h_t \odot f_t; h_t \odot J_t] \tag{7.4}$$

$$g_t = W_s^\top[k_t; J_t^d] \tag{7.5}$$

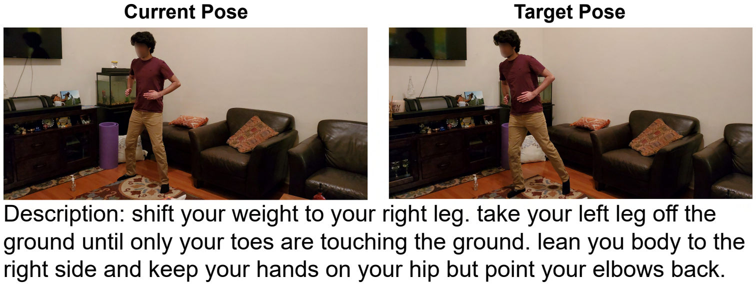
where Att is general attention (see Appendix A.11 for details). The next token is: $w_t = \text{argmax}(g_t)$, and the loss is: $L_{ML} = -\sum_t \log p(w_t^*|w_{1:t-1}^*, f, J)$, where $w_t^*$ is the GT token.

**RL Training.** We apply the REINFORCE algorithm (Williams, 1992) to learn a policy $p_\theta$ upon the model pre-trained with the maximum likelihood approach: $L_{RL} = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)]$; $\nabla_\theta L_{RL} \approx -(r(w^s) - b)\nabla_\theta \log p_\theta(w^s)$, where $w^s$ is a description sampled from the model, $r(\cdot)$ is the reward function, and $b$ is the baseline. We employ the SCST training strategy (Rennie et al., 2017) and use the reward for descriptions from the greedy decoding (i.e., $b = r(w^g)$) as the baseline. We also employ CIDEr as the reward, following Rennie et al. (2017)'s observation (using CIDEr as a reward improves overall metric scores). We follow the mixed loss strategy setup (Wu et al., 2016; Paulus et al., 2018): $L = \gamma_1 L_{ML} + \gamma_2 L_{RL}$.

**Multilingual Parameter Sharing.** We implement the multilingual training setup by sharing parameters between English and Hindi models, except the parameters of word embeddings, description LSTMs, and final fully connected layers, making the total number of parameters substantially less than those needed for the separate two models summed.

### 7.5.2 Target Pose Retrieval Model

The current and target candidate images are encoded the same way as the captioning model. A bidirectional LSTM encodes the descriptions: $c = \text{BiLSTM}(\hat{w})$. Image features are aligned



Description: shift your weight to your right leg. take your left leg off the ground until only your toes are touching the ground. lean you body to the right side and keep your hands on your hip but point your elbows back.

Figure 7.7: An example from Sim-To-Real transfer dataset.

with description features via cross attention.

$$\tilde{c}^c, \tilde{f}^{t_i}, \tilde{c}^{t_i}, \tilde{f}^c = \text{CA-Stack}(c, f^c, c, f^{t_i}) \tag{7.6}$$

$$k_{1i} = \text{Self-Gate}([\tilde{c}^c; \tilde{c}^{t_i}; \tilde{c}^c \odot \tilde{c}^{t_i}]) \tag{7.7}$$

$$g_{1i} = \text{Self-Gate}([\tilde{f}^{t_i}; \tilde{f}^c; \tilde{f}^{t_i} \odot \tilde{f}^c]) \tag{7.8}$$

where $\odot$ is the element-wise product (see Appendix A.11 for details of the Self-Gate). For joints feature, we calculate the difference between the two joints set: $J^{dt_i} = W_j^\top (J^{t_i} - J^c)$; $J^{dc_i} = W_j^\top (J^c - J^{t_i})$. We apply the same process that the image features go through (i.e., Eq. 7.6-7.8) to get $k_{2i}$ and $g_{2i}$.

$$p_i = W_p^\top [k_{1i}; g_{1i}; k_{1i} \odot g_{1i}] \tag{7.9}$$

$$q_i = W_q^\top [k_{2i}; g_{2i}; k_{2i} \odot g_{2i}] \tag{7.10}$$

$$s_i = W_s^\top [p_i; q_i; p_i \odot q_i] \tag{7.11}$$

The score $s_i$ is calculated for each target candidate and the one with the highest score is considered as the predicted one: $\hat{t} = \text{argmax}([s_0; s_1; ...; s_9])$.

## 7.6 Experimental Setup

**Data Splits.** For the pose-correction-captioning task, we split the dataset into train/val-seen/val-unseen/test-unseen following Anderson et al. (2018b). We assign separate rooms to val-unseen and test-unseen splits for evaluating model's ability to generalize to unseen environments. The number of task instances for each split is 5,973/562/563/593 (train/val-seen/val-unseen/test-unseen) and the number of descriptions is 5,973/1,686/1,689/1,779. For the target-pose-retrieval task, we split the dataset into train/val-unseen/test-unseen. In this task, "unseen" means "unseen animations". We split the dataset by animations so that the task cannot be easily done by memorizing/capturing patterns of certain animations in the image pairs. After filtering for the target candidates (see Sec. 7.2), we obtain 4,227/1,184/1,369 (train/val-unseen/test-unseen) instances. See Appendix A.12.1 for the detailed room and animation assignments.

**Training Details.** We use 512 as the hidden size and 256 as the word embedding dimension. We use Adam (Kingma and Ba, 2015) as the optimizer. See Appendix A.12.3 for details.

**Metrics.** For the pose-correction-captioning task, we employ automatic evaluation metrics: BLEU-4 (Papineni et al., 2002), CIDEr (Vedantam et al., 2015), METEOR (Banerjee and Lavie, 2005), and ROUGE-L (Lin, 2004). Also, motivated by previous efforts towards more reliable evaluation (Wiseman et al., 2017; Serban et al., 2017; Niu and Bansal, 2019; Zhang et al., 2019; Sellam et al., 2020), we introduce new task-specific metrics to capture the important factors. Object-match counts correspondences of environment objects, body-part-match counts common body parts mentioned, and direction-match counts the (body-part, direction) pair match between the model output and the ground-truth (see Appendix A.12.4 for more information on direction-match). In the target-pose-retrieval task, we use the accuracy of the selection as the performance metric.

**Human Evaluation Setup.** We conduct human evaluation for the pose-correction-captioning task models to compare the output of the vision-only model, the language-only model, and the full vision+language model qualitatively. We sample 100 descriptions from each model (val-seen split), then asked crowd-workers to vote for the most relevant description in terms of the image pair, and for the one best in fluency/grammar (or 'tied'). Separately, to set the performance upper limit and to verify the effectiveness of our distractor choices for the target-pose-retrieval task, we conduct another human evaluation. We sample 50 instances from the target-pose-retrieval

| Language | Models | Automated Metrics |  |  |  | Task-Specific Metrics |  |  | Human Eval. |  |
|---|---|---|---|---|---|---|---|---|---|---|
|  |  | B4 | C | M | R | object-match | body-part-match | direction-match | R | F/G |
| English | V-Only | 6.90 | 6.41 | 16.78 | 30.09 | 0.04 | 1.01 | 0.05 | 4% | 4% |
|  | L-Only | 17.74 | 11.42 | 22.14 | 35.16 | 0.08 | 1.22 | 0.15 | 15% | 27% |
|  | V+L | 17.55 | 14.47 | 21.29 | 35.21 | 0.18 | 1.29 | 0.13 | 48% | 45% |
| Hindi | V-Only | 8.43 | 4.37 | 18.90 | 28.55 | 0.03 | 1.21 | 0.02 | 9% | 10% |
|  | L-Only | 25.42 | 11.41 | 29.68 | 36.90 | 0.0 | 1.42 | 0.07 | 19% | 26% |
|  | V+L | 18.99 | 8.58 | 29.26 | 34.73 | 0.08 | 1.63 | 0.10 | 51% | 53% |

Table 7.2: The performance of the unimodal and multimodal models on automated metrics, our new task-specific metrics, and human evaluation. for both English and Hindi dataset on the val-seen split (B4: BLEU-4, C: CIDEr, M: METEOR, R: ROUGE, V: Vision+Joints, L: Language, R: Relevancy, F/G: Fluency and Grammar).

test-unseen split and ask an expert to perform the task for both English and Hindi samples. See Appendix A.12.2 for more details.

**Unimodal Model Setup.** We implement unimodal models (vision-/language-only) for comparison with the multimodal models. See Appendix A.12.5 for more details.

**Other Image-Difference Datasets.** We also evaluate our baseline model on other image-difference datasets to show that the baseline is strong and competitive: Image Editing Request (Tan et al., 2019a), NLVR2 (Suhr et al., 2019) (the variant from Tan et al. (2019a)), and CLEVR-Change (Park et al., 2019).

**Sim-to-Real Transfer.** To verify the possibility of the transfer of our simulated image dataset to real images, we collect real image pairs of current and target poses. We randomly sample 60 instances from test-unseen split (test-sim) and then the authors and their family members[4] follow the poses in the sampled test-sim split to create the real image version (test-real). Since the environments (thus objects and their layout too) and poses (though they are told to try to match as accurately as possible) have differences between the two splits (i.e., test-sim and test-real), we manually re-write a few words or phrases in the descriptions to make it more consistent with images in the test-real split (see Fig. 7.7).

[4] Hence covering diverse demographics, including some that are different from the simulator data splits, as well as different room environments. All participants consented to the collection of images (and additionally, we blur all faces).

| Language | Models | B4 | C | M | R |
|---|---|---|---|---|---|
| English | V+L | 17.55 | 14.47 | 21.29 | 35.21 |
| | (-) Joints | 17.39 | 13.79 | 21.35 | 34.86 |
| | (+) RL | 18.69 | 16.04 | 22.35 | 36.18 |
| | (+) Multi-L | 19.08 | 15.71 | 22.47 | 36.46 |
| Hindi | V+L | 18.99 | 8.58 | 29.26 | 34.73 |
| | (-) Joints | 18.23 | 7.93 | 27.55 | 34.12 |
| | (+) RL | 18.57 | 9.63 | 28.83 | 34.76 |
| | (+) Multi-L | 18.67 | 9.77 | 29.05 | 34.74 |

Table 7.3: Model ablations on val-seen split (RL: reinforcement learning, Multi-L: multilingual).

| Dataset | Model | B4 | C | M | R |
|---|---|---|---|---|---|
| Image Editing Request<br>Tan et al. (2019a) | DRA | 6.72 | 26.36 | **12.80** | 37.25 |
| | Ours | **7.88** | **27.70** | 12.53 | **37.56** |
| NLVR2<br>Suhr et al. (2019) | DRA | 5.00 | **46.41** | 10.37 | **22.94** |
| | Ours | **5.30** | 45.09 | **10.53** | 22.79 |
| CLEVR-Change (SC)<br>Park et al. (2019) | DUDA | 42.9 | 94.6 | 29.7 | - |
| | Ours | **44.0** | **98.7** | **33.4** | 65.5 |

Table 7.4: Our baseline V+L model performs competitively on other image-difference captioning datasets (DRA: Dynamic Relation Attention (Tan et al., 2019a), DUDA: Dual Dynamic Attention Model (Park et al., 2019); SC = Scene Change).

## 7.7 Results

### 7.7.1 Pose Correctional Captioning Task

As shown in Table 7.2, the V+L models show better performance than V-only models. The L-only model shows higher scores on some of the automatic metrics, likely because the descriptions in our FixMyPose dataset are instructional about body parts (and their movements/directions), so similar phrases are repeated and shallow metrics will only focus on such phrase-matching, not correctly reflecting human evaluations (Belz and Reiter, 2006; Reiter and Belz, 2009; Scott and Moore, 2007; Novikova et al., 2017; Reiter, 2018). Thus, we also evaluate the output of each model on our task-specific metrics that account the important factors (objects, body parts, and movement directions), and we also conduct human evaluation to check the real quality of the outputs. The V+L models show better performance on the task-specific metrics and human evaluation, meaning they capture essential information and their outputs are more relevant to

| Character No. | B4 | C | M | R |
|---|---|---|---|---|
| 1 | 20.23 | 9.44 | 21.87 | 35.98 |
| 2 | 17.54 | 7.43 | 20.20 | 34.70 |
| 3 | 18.54 | 7.24 | 20.74 | 35.58 |
| 4 | 19.00 | 9.28 | 20.43 | 34.01 |
| 5 | 19.77 | 10.59 | 21.08 | 35.01 |
| 6 | 20.28 | 7.94 | 20.94 | 35.47 |

Table 7.5: The V+L model's performance (English) on the individual characters' demographics. The balanced scores indicate that our dataset is not biased towards any specific demographic.

| Split | Automated Metrics | | | | Task-Specific Metrics | |
|---|---|---|---|---|---|---|
| | B4 | C | M | R | OM | DM |
| test-sim | 16.93 | 9.91 | 21.79 | 35.08 | 0.04 | 0.20 |
| test-real | 13.01 | 7.12 | 21.40 | 33.05 | 0.07 | 0.11 |

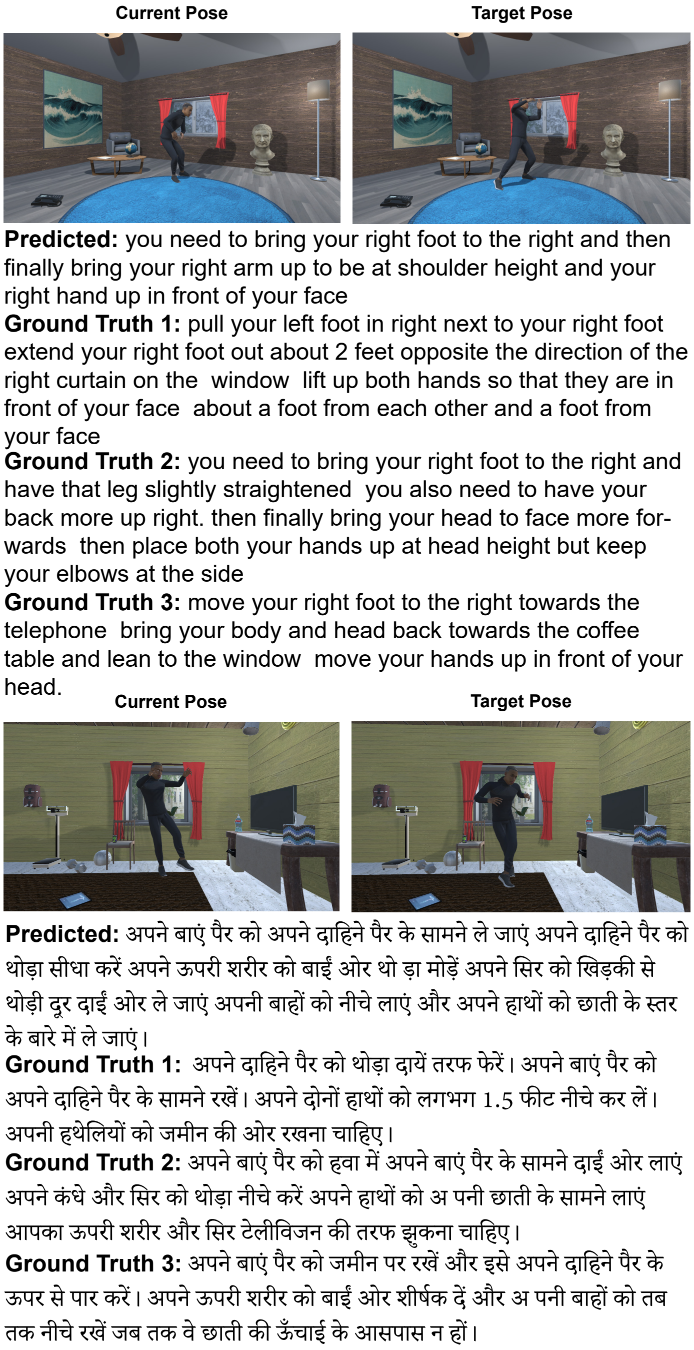
Table 7.6: Sim-to-Real transfer performance. Since there is no GT joints for real images, the body-part-match metric is not available (OM: object-match, DM: direction-match).

the images and more fluent in the respective language. See Appendix A.13.2 for "unseen" split results.[5]

**Ablations.** As Table 7.3 shows, adding body joints features improves the score much, implying body joints gives additional important information to capture human movements.

**RL/Multilingual Model Results.** As Table 7.3 shows, RL training helps improve scores by directly using the evaluation metric (CIDEr) as the reward. We leave exploring more effective reward functions (e.g., the joints distance from a reverse pose generation task) for future work. Table 7.3 also shows that the multilingual training setup achieves comparable scores (similar observation to Wang et al. (2019b)) with only 71% of the parameters of the separate training setup (13.2M vs 18.7M), promising future work on more compact and efficient multilingual models.

[5] We also checked for variance by running models with 3 different seeds and the stddev is small (less than/near 0.5% on CIDEr).



**Predicted:** you need to bring your right foot to the right and then finally bring your right arm up to be at shoulder height and your right hand up in front of your face
**Ground Truth 1:** pull your left foot in right next to your right foot extend your right foot out about 2 feet opposite the direction of the right curtain on the window lift up both hands so that they are in front of your face about a foot from each other and a foot from your face
**Ground Truth 2:** you need to bring your right foot to the right and have that leg slightly straightened you also need to have your back more up right. then finally bring your head to face more forwards then place both your hands up at head height but keep your elbows at the side
**Ground Truth 3:** move your right foot to the right towards the telephone bring your body and head back towards the coffee table and lean to the window move your hands up in front of your head.

**Predicted:** अपने बाएं पैर को अपने दाहिने पैर के सामने ले जाएं अपने दाहिने पैर को थोड़ा सीधा करें अपने ऊपरी शरीर को बाईं ओर थो ड़ा मोड़ें अपने सिर को खिड़की से थोड़ी दूर दाईं ओर ले जाएं अपनी बाहों को नीचे लाएं और अपने हाथों को छाती के स्तर के बारे में ले जाएं।
**Ground Truth 1:** अपने दाहिने पैर को थोड़ा दायें तरफ फेरें। अपने बाएं पैर को अपने दाहिने पैर के सामने रखें। अपने दोनों हाथों को लगभग 1.5 फीट नीचे कर लें। अपनी हथेलियों को जमीन की ओर रखना चाहिए।
**Ground Truth 2:** अपने बाएं पैर को हवा में अपने बाएं पैर के सामने दाईं ओर लाएं अपने कंधे और सिर को थोड़ा नीचे करें अपने हाथों को अ पनी छाती के सामने लाएं आपका ऊपरी शरीर और सिर टेलीविजन की तरफ झुकना चाहिए।
**Ground Truth 3:** अपने बाएं पैर को जमीन पर रखें और इसे अपने दाहिने पैर के ऊपर से पार करें। अपने ऊपरी शरीर को बाईं ओर शीर्षक दें और अ पनी बाहों को तब तक नीचे रखें जब तक वे छाती की ऊँचाई के आसपास न हों।

Figure 7.8: Output examples of our multimodal model in English (top) and Hindi (bottom).

**Other Image-Difference Datasets.** Table 7.4 shows that our V+L baseline model beats or matches state-of-the-art models on other datasets, implying our baseline models are strong starting points for our FIXMYPOSE dataset.

**Output Examples.** Outputs from our V+L models are presented in Fig. 7.8. The English model captures the movement of the character's legs and arms ("bring your right foot to the right" and "bring your right arm up to be at shoulder height ... right hand up in front of your face"). The Hindi model captures movement of the body parts and their spatial relationship to each other (English translation: "move your left leg in front of your right leg..."), the model can also describe movement using object referring expressions (English translation: "...move your head slightly away from the window..."). See Fig. 7.8 for the original Hindi and Appendix A.13.1 for full analysis and unimodal outputs.

**Demographic Ablations.** We split the dataset into subsets for each individual character avatar, and evaluate our V+L model on each subset. As shown in Table 7.5, scores from each subset are reasonably balanced, indicating our dataset is not skewed to favor a specific demographic or character.

**Sim-to-Real Transfer.** As shown in Table 7.6, the sim-to-real performance drop is not large, meaning information learned from our simulated FIXMYPOSE dataset can be transferred to real images reasonably well. Also, considering that the results are from a set of images of people with different demographics and different environments, there is no particular bias in the models' output which is trained on our dataset. Since there is no GT body joints for the real images, we modify our model so it can also be trained to predict the joints during training time as well as generate descriptions (i.e., in a multi-task setup) and use the estimated joints at test time.[6]

### 7.7.2 Target Pose Retrieval Task

As shown in Table 7.7, V+L models show the highest scores for the target-pose-retrieval task, indicating that achieving high performance is not possible by exploiting unimodal biases. V-Only models score higher than the random-selection model, which selects an image at random,

[6] For the simulated data results in Table 7.3 (English), we obtain a CIDEr score of 14.17 using predicted joints (on the val-seen split), which as expected is between the non-joint (13.79) and GT-joint (14.47) models' results (hence showing that reasonable performance can be achieved without GT joint information at test time). The average distance between predicted and GT joints is around 0.4 meters.

| Models | Accuracy (%) | |
|---|---|---|
| | English | Hindi |
| Random-Selection | 9.81 | |
| V-Only | 34.82 | |
| L-Only | 8.86 | 8.96 |
| V+L | 38.49 | 37.84 |
| Human | 96.00 | 96.00 |

Table 7.7: The scores for the target-pose-retrieval task. While the V+L models scores the highest, there is still much room for improvement when compared with human performance.

because even with our careful distractor choices (see Sec. 7.2 and Appendix A.8), the poses in the "current" and "target" images are more similar to each other than the other images. However, the human-model performance gap is still quite large, implying there is much room for improvement.[7]

## 7.8 Conclusion

We introduced FixMyPose, a novel pose correctional description dataset in both English and Hindi. Next, we proposed two tasks on the dataset, pose-correction-captioning and target-pose-retrieval, both of which require models to understand diverse linguistic properties such as egocentric relation, environmental direction, implicit movement description, and analogous reference as well as capture fine visual movement presented in two images. We also presented unimodal and multimodal baselines as strong starter models. Finally, we demonstrated the possibility of transfer to real images. In future work, we plan to further expand the FixMyPose dataset with more languages and even more diversity in the character pool (e.g., height, age, etc. based on digital avatar availability) and animations.

[7] Human performance is 96% when given the full task (English), but much lower when only given lang. (38%) or only vis. (22%), further indicating that both lang.+vis. is needed to solve the task.

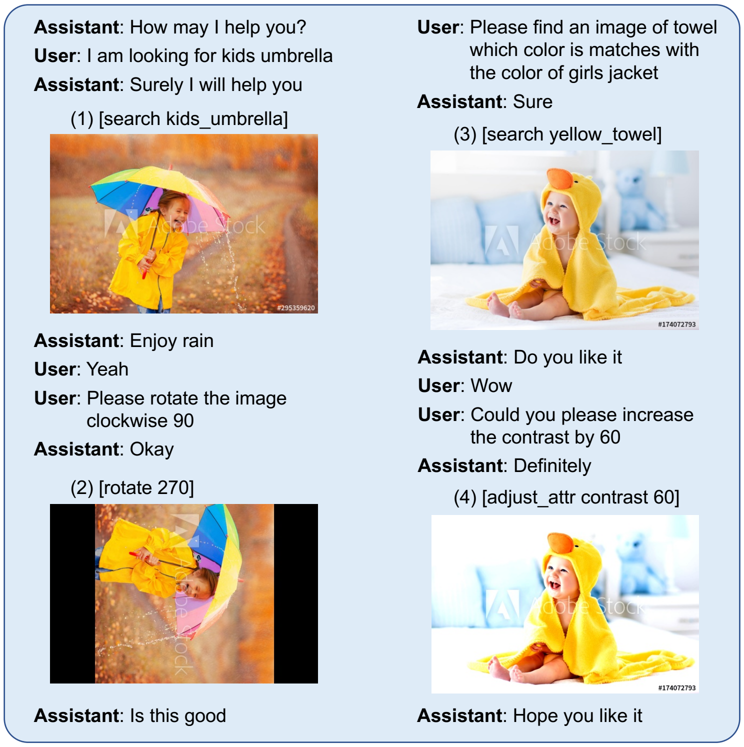
# CHAPTER 8: CONVERSATIONAL AGENT FOR IMAGE SEARCH AND EDITING

## 8.1 Introduction

As the technology of image editing is developing and being refined, its utility is also increasing. It has become a usual practice to add editing effects to photos to make them look better. However, using image editing tools requires the expertise and skill that regular layperson users do not have. The names of these photo effects are not familiar and even the implication of the effects on images are not intuitive for most users. Hence, to increase the accessibility to these tools, proper individual expert guidance is required. However, guidance assistant systems run by small groups of available human experts could not cover all the requests from a large number of users worldwide. Instead, editing tools can benefit from having an automated assistant system that can have a conversation with users at scale to help them with their editing needs.

On the other hand, as the purpose and use cases of image editing are getting diverse, source materials for image editing also need to be diversified. For example, users might want to recreate their photos by adding additional objects from external sources. Users may also want to follow a reference image to make their photos more attractive (e.g., by borrowing a color from the source image). Hence, these activities call for an image search interface to be integrated with image editing tools to provide a more integrated and comprehensive platform.

There have been some prior efforts towards automated image editing systems. They have focused on intent/action/goal identification from image editing requests (Manuvinakurike et al., 2018a,b,c; Lin et al., 2018), exploring low-level editing terms (Lin et al., 2020), editing images from descriptions (Shi et al., 2020), and describing image differences caused by image editing (Tan et al., 2019a). However, there has been limited effort to integrate conversational image search and editing functions in a directly executable end-to-end manner for deployment into



**Assistant**: How may I help you?

**User**: I am looking for kids umbrella

**Assistant**: Surely I will help you

(1) [search kids_umbrella]

**Assistant**: Enjoy rain

**User**: Yeah

**User**: Please rotate the image clockwise 90

**Assistant**: Okay

(2) [rotate 270]

**Assistant**: Is this good

**User**: Please find an image of towel which color is matches with the color of girls jacket

**Assistant**: Sure

(3) [search yellow_towel]

**Assistant**: Do you like it

**User**: Wow

**User**: Could you please increase the contrast by 60

**Assistant**: Definitely

(4) [adjust_attr contrast 60]

**Assistant**: Hope you like it

Figure 8.1: Conversational agent for image search and editing (CAISE). The dialogue starts with the image search request from the user. The assistant conducts the image search and addresses the following image search or editing requests for the user through 4 turns of request-execution exchange ([·] shows the image search/editing commands to the system).

real-world applications. Therefore, we propose a new dataset, CAISE (‘**C**onversational **A**gent for **I**mage **S**earch and **E**diting’), in which a user and an assistant hold a conversation in natural language (English) about image search and editing (Figure 8.1). The user’s role is to make requests for image search and editing and the assistant’s role is to search or edit images according to the user’s requests and return the results while responding with natural language.

To collect such data, we implement a dialogue interface and ask pairs of annotators (one operating as the user and the other one as the assistant) to converse and search/edit images via the interface. The user is provided with multiple seed images from which they can get some ideas about what to search in the first place. Also, we show the user a list of suggested image search/editing functions to keep the command types diverse by asking them to follow the list as long as they can. The assistant annotator, on the other hand, is equipped with an image search and editing interface to perform the user’s requests. All command executions lead to the corre-

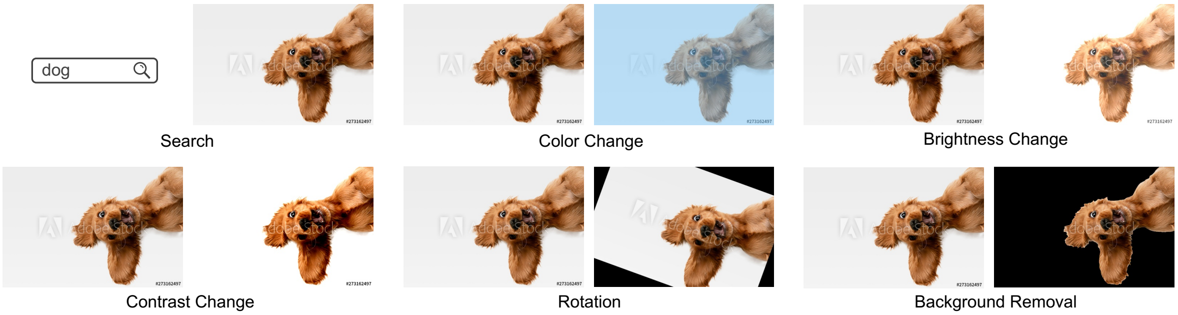


Figure 8.2: The diverse image search and editing effect functions that our CAISE dataset employs.

sponding executable commands to be recorded. A total of 1.6K dialogues and 6.2K task instances are collected. The collected dialogues contain different types of image search/editing requests from users (direct request, implied request, object referring request; Table 8.3) and assistants' diverse responses (Sec.8.4), requiring models to understand the diverse grounded interactions in the conversations.

The task on the CAISE dataset is to generate the executable commands (e.g., search, color-change, brightness-change, contrast-change, rotation, background-removal; Figure 8.2) given the conversation and image contexts. This task setup simulates real-world image editing tools, facilitating important initial steps towards deployment in downstream applications. We introduce a novel generator-extractor model as a strong starting point baseline for this task and dataset. We employ a copying mechanism (Vinyals et al., 2015; Gu et al., 2016; Miao and Blunsom, 2016; See et al., 2017), with which the model adaptively selects a way (i.e., generate from the vocabulary or extract from the context) to decode the next word since the clues for arguments of an executable command could be implicitly mentioned in the user's request (e.g., *"Please change the image color with* ***color of bus****"*) or the request contains the direct cues (e.g., *"Is it possible to increase the brightness of the image by* ***30*** *percent"*). For more effective model performance, we extend this mechanism so that it can also cover visual concepts in images by extracting object attributes or names from a set of object detection based concepts. For example, for the request *"Please change the image color with the color of bus"*, the corresponding color can not only be generated from the vocabulary, but also directly copied from one of object detection results, *"red* *bus"*. Our experiments show our baseline model performs effectively as a starting point, and we also demonstrate a large human-machine performance gap to allow useful future works on this important and understudied task.

Our contributions are two-fold: (1) we introduce a novel grounded dialogue dataset, CAISE, which incorporates image search and editing, featuring executable commands, hence allowing for more practical use in real-world applications. (2) We also introduce a generator-extractor model as a strong starting point baseline which extends the copy mechanism to the visual concept extraction, allowing for more effective performance and helping the interpretation of the model's behavior, while also leaving a large human-machine performance gap to allow useful future work by the community on this new challenging multimodal task.

### 8.2 Task Description

Multimodal dialogue based executable command generation is one task that can be introduced from our CAISE dataset. Specifically, given a conversation history, previously searched and edited images, and previously executed commands, the agent should generate an executable command which can return the correct result for the user's request. The definitions of the executable commands are as follows:

**Search.** The search command retrieves images that are searched online with a query string. The format of the search command is [*search argument_1 ... argument_n ...*]. *'argument_n'* is the n-th token in the query string and there is no limit for the number of arguments.

**Color Change.** The color change command paints a whole image with a designated color. The format of the color change command is [*adjust_color argument_1 argument_2*]. *'argument_1'* is a name of the colors (red, orange, green, blue, sky blue, purple, brown, yellow, pink), and *'argument_2'* is the value of intensity (0.0-1.0).

**Brightness Change.** The brightness change command changes the brightness of a whole image with a designated intensity. The format of the brightness change command is [*adjust_attr brightness argument_1*]. *'argument_1'* is the value of intensity (-100-100%).

**Contrast Change.** The contrast change command changes the contrast of a whole image with a designated intensity. The format of the contrast change command is [*adjust_attr contrast argument_1*]. *'argument_1'* is the value of intensity (0-100%).

**Rotation.** The rotation command rotates a whole image by a designated degree. The format of the rotation command is [*rotate argument_1*]. *'argument_1'* is the value of degree (0-360).

**Background Removal.** The background removal command makes a whole image black except the main subject. The format of the background removal command is [*image_cutout*]. There is no argument.

For illustrations of these photo effects, see Figure 8.2.

### 8.3 Dataset

Our CAISE dataset consists of conversations between a 'user' and an 'assistant'. Each conversation includes utterances of the user and assistant, searched or edited images, and executed commands.

**Conversation Interface.** We implement a dialogue system through which a pair of people chat about image search and editing. We build the user-side and the assistant-side interfaces separately since their roles are quite different. In the user-side interface, we provide 15 random seed images from COCO dataset (Lin et al., 2014) to help the user decide what to request for the first image search. We also present a suggestion for types of search and editing, which is a list of four commands from different types being randomly selected and ordered to avoid repeating the same search/editing order so that the user can follow it when they request to the assistant. In the assistant-side interface, we prepare a customized light-weight search and editing tool for the assistant to address the users' requests. We use Adobe Stock[1] for the image search engine, Adobe Photoshop[2] for the background removal function, and OpenCV[3] to implement the other editing functions. All the search and editing effects conducted from the tool are recorded in the form of executable commands that are used for corresponding functions. See Appendix A.14 for the images of the interfaces.

**Data Collection.** We employ 10 annotators and train them to make them familiar with the collection interfaces and their primary roles, and guarantee the quality of the dataset. In the training session, we check all the practice dialogues manually and give feedback. We perform this training session multiple times until the quality of the dialogues gets above some threshold (see Appendix A.15 for the detailed training process). After the training period, two annotators are paired so that one of them takes the user role and the other takes the assistant role. User-annotators are asked to give four requests throughout a conversation. Assistant-annotators are asked to perform the image search and editing functions according to the user-annotators' requests. If the user-annotators' requests are not clear, the assistant-annotators can ask them to clarify. We hire freelancers since the collection process needs some training to build expertise (especially for manipulating the search/editing interface), and pairing between the user and assistant annotators via a general crowd-sourcing platform is not easy.[4]

**Payment.** We pay up to 2 USD per dialogue, including bonuses. We also pay for dialogues which are created by annotators in their training period. Considering the time taken for a dialogue (around 5 minutes for a pair of trained annotators), the hourly wage is competitive (nearly 12 USD/hour per annotator).

[1] `https://www.adobe.io/apis/creativecloud/stock.html` (the watermarks on the images are from using the Adobe Stock API).

[2] `https://adobedocs.github.io/photoshop-api-docs-pre-release/`

[3] `https://opencv.org/`

[4] We use Upwork (`https://www.upwork.com`) to hire freelancer annotators for high-quality, trained-expert human feedback. Upwork provides various communication tools (text chat and video/audio call interfaces) to facilitate communication with annotators and thus enable effective and efficient annotator training, as also shown in (Stiennon et al., 2020).

| | Count | |
|---|---|---|
| | Per Dialogue | Total |
| Dialogue | - | 1,611 |
| Utterance | 15.5 | 24,938 |
| Utterance (user) | 7.9 | 12,641 |
| Utterance (assistant) | 7.6 | 12,297 |
| Executable Command | 3.8 | 6,173 |
| Image | 3.8 | 6,173 |

Table 8.1: The number of dialogue components. Dialogues in our CAISE dataset are long (15.5 utterances) with four turns of image search/editing request-execution exchanges.

| | Length | | | | |
|---|---|---|---|---|---|
| | avg | stddev | median | max | min |
| Utterance | 5.26 | 4.98 | 4.0 | 38 | 1 |
| Utterance (user) | 6.99 | 6.16 | 6.0 | 38 | 1 |
| Utterance (assistant) | 3.49 | 2.24 | 3.0 | 24 | 1 |

Table 8.2: The lengths of utterances in the dialogue collection. The user utterances are longer than assistant's due to the difference in their roles. The standard deviation of the lengths is large, indicating the utterances have various lengths.

## 8.4 Data Analysis

We collect 1,611 dialogues and create 6,173 task instances from the dialogue collection (since each dialogue has around four executable commands).

**Dialogue Length.** As shown in Table 8.1, the average number of utterances from both the users and assistants is 15.5 (7.9 and 7.6 from users and assistants, respectively). The average number of executable commands and images are the same (3.8 per dialogue) since each image is the result of the execution of each corresponding command.

**Utterance Length.** As shown in Table 8.2, the average length of user utterances is larger than that of assistant utterances (6.99 vs. 3.49). The reason is that user utterances are mainly about image search and editing requests, requiring detailed explanations (e.g., *"Could you also find*

| Type | Examples |
|---|---|
| Dir-Req | *"I was looking for an image of zoo"*<br>*"Now increase the brightness of the image by 40 percent"*<br>*"Please get rid of the background"* |
| Impl-Req | *"Can we repeat further by 130 degree more"*<br>*"Can we try increasing further by 50 more"* |
| ObjRef-Req | *"Please find an image of the object seen to the right of the juicer in the above image"*<br>*"Please change the color of image which matches with the color of cushion"* |

Table 8.3: The examples of different types of requests (Dir-Req: direct request, Impl-Req: implied request, ObjRef-Req: object referring request).

*me an image of dress for my wife matching the color of hat in the above image?"*). On the other hand, assistant utterances are usually short responses to users' requests (e.g., *"okay"*, *"sure"*) or questions for users' confirmation (e.g., *"Do you like it?"*, *"Is this fine?"*), and clarifications (e.g., *"clock wise or anti clockwise?"*). Also, the standard deviations of the utterance lengths are large compared to the average lengths, confirming utterances in our CAISE dataset have various lengths.

**User Request Types.** As shown in Table 8.3, we can categorize the image search and editing requests mainly into three types: direct request, implied request, and object referring request. Direct requests are self-contained requests which have direct clues about what users are asking. Implied requests are the ones that do not explicitly mention what types of functions are asked to be performed but imply them from the conversation contexts. Object referring requests are the ones that use the information of objects (i.e., color, name, location) in images to specify what should be done.

**Assistant Response Types.** Although several assistant responses are generic confirmation-based (since the assistants' main role is to perform image search and editing according to users' requests), there are also several other types of interesting responses such as correction (user: *"... image of sea-saw ..."* - assistant: *"Do you mean see-saw?"*), ambiguity-clarification (user: *"...*

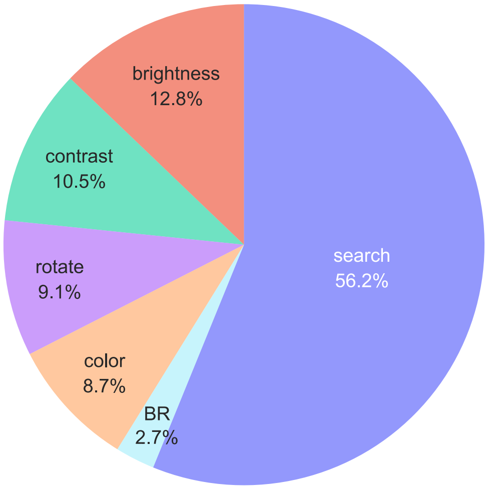
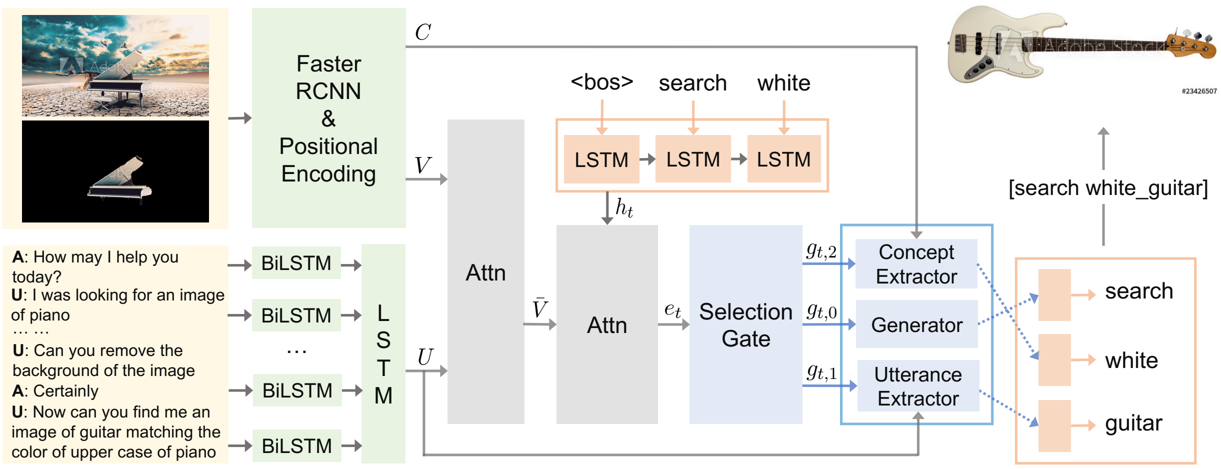


Figure 8.3: The executable commands frequency. The search command has the highest frequency since each dialogue begins with a search request (BR: background removal).

*rotate the image to 40 degree"* - assistant: *"... clock wise or anti clockwise?"*), coreference (assistant: *"How would you like it by"*), etc., encouraging models to understand the diverse grounded interactions in the conversations between users and assistants to perform the task.

**Executable Commands Frequency.** As shown in Figure 8.3, the search command is the most frequent. The reason is that search is the first command that must be performed in every dialogue, and we design the collection interface so that each dialogue has one additional search command on average (the ratio of the first-line search commands to the other search commands is 46.5% vs. 53.5%). The low frequency of the background removal command ("BR" in the figure) is due to the command's instability. Unlike the other commands that do not fail, the background removal command could fail depending on images' contents (it seems that images that have complicated contents are hard to remove background from). Once the background removal command fails, the user-annotators might not try it again and perform one of the other functions instead.



**A**: How may I help you today?
**U**: I was looking for an image of piano
… …
**U**: Can you remove the background of the image
**A**: Certainly
**U**: Now can you find me an image of guitar matching the color of upper case of piano

Figure 8.4: The Generator-Extractor model. The model adaptively selects the source of the next token via the selection gate (for the simplicity purpose, some blocks and relationship arrows are omitted; input "<bos> search white" to the LSTM block is previously generated tokens, i.e., an autoregressive decoding setup; the model produces the command word-by-word).

## 8.5 Models

We present the generator-extractor model as a starting point baseline (Figure 8.4). The model takes the history of utterances, images, and previously executed commands as input, and predicts a next executable command.

**Encoder.** A large part of our CAISE dataset involves objects and their concepts (names and attributes) in images, especially for the search command. Thus, we employ Faster R-CNN (Girshick, 2015) to extract object visual features $\hat{V}$, bounding box features $B$, and their concept features $W^c$, which are usually made of a couple of tokens, from images $I$. $\hat{V}$ and $B$ are combined through a linear layer, and $W^c$ is further encoded by a word embedding layer and the bidirectional LSTM (Hochreiter and Schmidhuber, 1997):

$$\hat{V}, B, W^c = \text{FRCNN}(I), \quad V = \text{PE}(\text{Linear}([\hat{V}; B])) \tag{8.1}$$

$$\hat{C} = \text{Embed}(W^c), \quad C = \text{PE}(\text{BiLSTM}(\hat{C})) \tag{8.2}$$

where PE denotes positional encoding (Gehring et al., 2017; Vaswani et al., 2017) and it is applied image-wise (i.e., the same encoding value is applied to the features from the same image).

Sequences of tokens from utterances $W^u$ in dialogue $D$ are encoded by the bidirectional LSTM, and the last forward hidden state and the first backward hidden state of $\hat{U} \in \mathbb{R}^{M \times N \times d}$ are extracted and concatenated to create a vector which represents each utterance, where $M$ is the dialogue length, $N$ is the utterance length, and $d$ is the feature dimension. Then, the sequence of the utterance features is fed to a LSTM to learn the dialogue context:

$$\hat{U} = \text{BiLSTM}(\text{Embed}(W^u)) \tag{8.3}$$

$$U = \text{LSTM}([\hat{U}^f_{N-1}; \hat{U}^b_0]) \tag{8.4}$$

We employ the attention mechanism to align the visual features $V$, and utterance features $U \in \mathbb{R}^{M \times d}$. We calculate the similarity matrix $S \in \mathbb{R}^{O \times M}$ between visual and utterance features, where $O$ is the total number of all object features from the images: $S_{ij} = V_i^{\top} U_j$. From the similarity matrix, the new fused visual and utterance feature is:

$$\bar{U} = \text{softmax}(S) \cdot U, \quad \bar{V} = [V; \bar{U}; V \odot \bar{U}] \cdot W_v \tag{8.5}$$

where $W_v \in \mathbb{R}^{3d \times d}$ is the trainable parameter, $\odot$ is element-wise product, and $\cdot$ is matrix multiplication. Tokens from an executable command, $\{w_t\}_{t=1}^{T}$, are embedded in the embedding layer, and then sequentially fed to the LSTM layer.

$$\hat{w}_{t-1} = \text{Embed}(w_{t-1}), \quad h_t = \text{LSTM}(\hat{w}_{t-1}, h_{t-1}) \tag{8.6}$$

The same (but with different parameters) attention mechanism (Attn), which is applied to visual and utterance features, is used for aligning the command feature, $h_t$, and $\bar{V}$.

$$e_t = \text{Attn}(h_t, \bar{V}) \tag{8.7}$$

**Generator.** The generator calculates the probability of each token in the vocabulary that contains all possible candidates.

$$l_t = \text{Linear}(e_t), \quad a_t^g = \text{softmax}(l_t) \tag{8.8}$$

**Extractor.** Utterances in our CAISE dataset contain many direct clues for generating commands. Thus, the model would benefit from extracting keywords from the context. We employ a copying mechanism (Vinyals et al., 2015; Gu et al., 2016; Miao and Blunsom, 2016; See et al., 2017) to implement the extraction.

$$(A_t^u)_i = h_t^\top U_i, \quad a_t^u = \text{softmax}(A_t^u) \tag{8.9}$$

The model also can directly obtain useful information from the visual concept since visual concept features can provide object names/attributes in a textual semi-symbolic format.

$$(A_t^c)_i = e_t^\top C_i, \quad a_t^c = \text{softmax}(A_t^c) \tag{8.10}$$

**Selection Gate.** To adaptively select the source of the next token, we employ gating approach (See et al., 2017) to obtain the adaptive weights:

$$g_t = \text{softmax}(W_g^\top e_t) \tag{8.11}$$

where $W_g \in \mathbb{R}^{d \times 3}$ is the trainable parameter. The weighted sum of each probability from each source is the final probability of the next token.

$$p(w_t | w_{1:t-1}, I, D) = g_{t,0} \cdot a_t^g + g_{t,1} \cdot a_t^u + g_{t,2} \cdot a_t^c \tag{8.12}$$

The loss is:

$$L = -\sum_{t=1}^{T} \log p(w_t^* | w_{0:t-1}, I, D) \tag{8.13}$$

where $w_t^*$ is the GT token.

## 8.6 Experiments

**Data Splits.** We split the total 1,611 dialogues into 1,052, 262, and 297 for train, validation, and test set, respectively. From the dialogue splits, we obtain 4,059/1,002/1,112 (train/valid/test) instance splits.

**Evaluation Metric.** We use accuracy as the evaluation metric. For image search and editing systems, it is important to feed the correct command, and automatic metrics for text generation tasks such as BLEU (Papineni et al., 2002) are not appropriate. So, we only count generated commands which exactly match the ground-truth commands (i.e., command types and their arguments) as the correct ones. For the search command, generated commands with different query word orders (e.g., [*search juice glass*] and [*search glass juice*]) are also considered correct since queries with different word orders usually return the same or similar outcomes. For the color change command, we only compare the command type and color names but not up to intensity (e.g., [*adjust_color blue*]) since, in most cases, users ask to change colors without saying a specific value of intensity (e.g., *"Color the image to the same color as the salmon in the above image"*).

**Human Expert Performance.** We randomly sample 50 instances for the search command and 10 instances for each of the other commands (total 100 instances) and ask an expert who knows the task well to predict the commands based on the textual and visual context.

**Training Details.** We use 512 as the hidden size and 256 as the word embedding dimension. We use Adam (Kingma and Ba, 2015) as the optimizer with the learning rate $1 \times 10^{-4}$. See Appendix A.16 for more details.

| | Models | Accuracy (%) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | total | search | color | brightness | contrast | rotation | remove-back |
| 1 | Base | 22.33 | 11.45 | 28.63 | 32.13 | 46.46 | 9.52 | **100.0** |
| 2 | Base+VE | 22.12 | 11.00 | 30.77 | 30.12 | 48.56 | 8.93 | **100.0** |
| 3 | Base+UE | 45.23 | 36.42 | 26.07 | **49.80** | 92.13 | **29.17** | 97.14 |
| 4 | Base+UE+VE | **46.43** | **37.43** | **40.60** | 48.39 | **93.18** | 26.49 | 97.14 |
| 5 | Human Expert | 90.0 | 82.0 | 90.0 | 100.0 | 100.0 | 100.0 | 100.0 |

Table 8.4: Model performance on the test split. The extractors help improve the model's performance (Base: the basic encoder-decoder model only with generator (without extractors), UE: utterance extractor, VE: visual concept extractor).

### 8.7 Results

As shown in Table 8.4, the extractor modules help improve the model's performance. The utterance extractor helps much to improve the model's performance (row 1 and 3). Especially, the scores for the search, brightness change, contrast change, and rotation commands get increased, implying that the utterance extractor can effectively locate the direct clues from the dialogue history context. Applying the visual concept extractor additionally increases the score (rows 3 and 4).[5] The performance of the search and color change commands gets improved from this application, meaning that the visual concept extractor can match the visual features and the concept features, and align them with requests. But, when comparing rows 1 and 2, adding the visual concept extractor to the base model does not seem to help. Although it shows a similar improvement pattern for the other commands, the performance for the search command is not improved. That implies that the visual concept extractor is effective together with the utterance extractor (see the example at the top of Figure 8.5).[6]

**Human Expert Performance.** As shown in row 4 and 5 of Table 8.4, the human-machine performance gaps are large for most of the command types, implying that there is large room for future

[5] The stddev of the full model (Base+UE+VE) scores is 0.74, and the score of the model on validation split is 49.7%.

[6] While we evaluate the performance via the average score over each search/editing instance like in (Das et al., 2017), one other possible evaluation option for practical applications is to consider the average success rate of the whole search/editing dialogue (5.4% from our full (Base+UE+VE) model).

|   | Models | Accuracy (%) |
|---|---|---|
| 1 | Request-Only | 42.30 |
| 2 | DialogHistory-Only | 0.66 |
| 3 | Request+DialogHistory | 43.17 |
| 4 | Vision-Only | 0.93 |
| 5 | Request+Vision | 45.56 |
| 6 | Request+DialogHistory+Vision | 46.43 |

Table 8.5: Modality ablations. Each modality/component helps improve the model's performance.

work to develop novel improvements on this new multimodal dialogue task, and our baseline described above is meant to serve as a strong starting point.

**Modality Ablation.** Table 8.5 shows the ablation results from different combinations of the model (Base+UE+VE) components. We take the last two utterances from the dialogue as 'request' since there is no explicit division between request and context in our CAISE dataset. As shown in row 2 and 4, the model could not perform well without the 'request'. That is obvious since, without this information, the model cannot figure out what and how to search and edit. The request-only (row 1) records a high score possibly because many of the requests contain direct clues like *"Can you rotate the image counterclockwise by 30 degrees"*. Adding dialogue history (row 1 and 3, row 5, and 6) helps, meaning the request needs dialogue context for better performance. Also, adding visual context (images) improves the model's performance (row 1 and 5, 3 and 6) because there are requests (such as for the search and color change commands) that need to refer to objects/colors in the visual context to be performed correctly.[7]

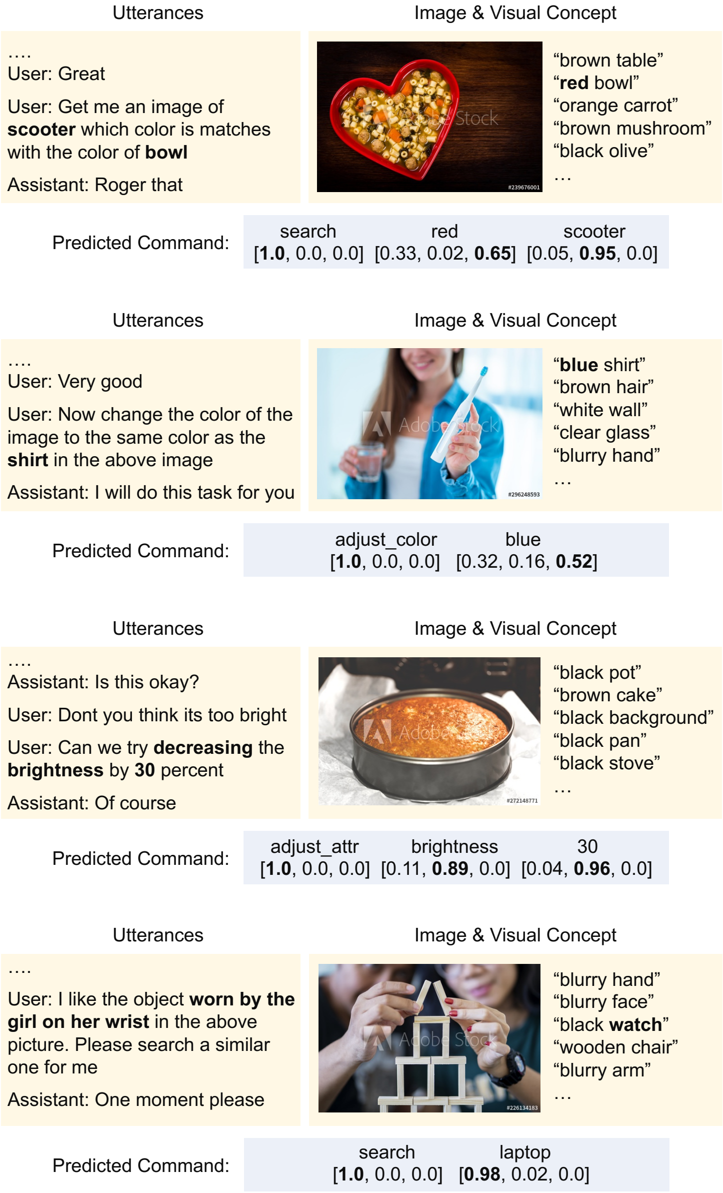
**Output Examples.** Figure 8.5 shows examples of the model output. In the top figure, our model gives the correct command ([*search red scooter*]) according to the request. Specifically, the model generates the command name, *'search'*, using the generator (with the selection gate weight

[7] We randomly sample 75 instances (except the first-turn search command) and conduct human evaluation on which inputs are required to perform the requests. Request-only: 43%; need-DialogHistory+Vision: 57% (need-DialogHistory 13%, need-Vision 47%, need-both 3%). This means that to solve our command generation task, models need to understand the context (we observe the same trend when we also include the first search command).

of 1.0), extracts the color, *'red'*, using the visual concept extractor (with the weight of 0.65), and also extracts the item name to search for, *'scooter'* using the utterance extractor (with the weight of 0.95). The second figure shows the example of the color change command. The model also generates the correct command name, *'adjust_color'* using the generator (with the weight of 1.0). The model then extracts the color, *'blue'*, from the visual concept feature using the visual concept extractor (with the weight of 0.52). On the other hand, as shown in the third figure, the model fails to understand the meaning of *'decreasing'* and just extracts *'30'* using the utterance extractor (with the weight of 0.96) for intensity (the ground-truth value is -30). In the bottom figure, the model cannot catch *'watch'* in the image and generated the wrong searching query, *'laptop'* using the generator (with the weight of 0.98). These negative results from our baseline model imply that there is room for improvement via more advanced modeling approaches in future work from the community on our CAISE task.

### 8.8 Conclusion

We introduced a novel conversational image search and editing task/dataset, called CAISE, in which an agent should conduct image search and editing according to users' requests. To implement and train the automated system, we collected a dialogue dataset in which a user and an assistant hold a conversation on image search/editing. We presented the generator-extractor model as a strong starting point baseline and the large human-machine performance gap showed there is room for improvement on this task.



….
User: Great

User: Get me an image of **scooter** which color is matches with the color of **bowl**

Assistant: Roger that

“brown table”
“**red** bowl”
“orange carrot”
“brown mushroom”
“black olive”
…

| search | red | scooter |
|---|---|---|
| [**1.0**, 0.0, 0.0] | [0.33, 0.02, **0.65**] | [0.05, **0.95**, 0.0] |

….
User: Very good

User: Now change the color of the image to the same color as the **shirt** in the above image

Assistant: I will do this task for you

“**blue** shirt”
“brown hair”
“white wall”
“clear glass”
“blurry hand”
…

| adjust_color | blue |
|---|---|
| [**1.0**, 0.0, 0.0] | [0.32, 0.16, **0.52**] |

….
Assistant: Is this okay?

User: Dont you think its too bright

User: Can we try **decreasing** the **brightness** by **30** percent

Assistant: Of course

“black pot”
“brown cake”
“black background”
“black pan”
“black stove”
…

| adjust_attr | brightness | 30 |
|---|---|---|
| [**1.0**, 0.0, 0.0] | [0.11, **0.89**, 0.0] | [0.04, **0.96**, 0.0] |

….

User: I like the object **worn by the girl on her wrist** in the above picture. Please search a similar one for me

Assistant: One moment please

“blurry hand”
“blurry face”
“black **watch**”
“wooden chair”
“blurry arm”
…

| search | laptop |
|---|---|
| [**1.0**, 0.0, 0.0] | [**0.98**, 0.02, 0.0] |

Figure 8.5: The examples of the model output (1st and 2nd examples: correct / 3rd and 4th: incorrect). Our model can effectively use the generator and extractors by selecting them with the adaptive selection gate (the numbers in bracket are the selection gate weight, i.e., [weight for the generator, weight for the utterance extractor, weight for the visual concept extractor]). The bottom two figures show incorrect examples in which the model cannot figure out the meaning of *‘decreasing’* and cannot catch *‘watch’* from the image.

# CHAPTER 9: SUMMARY AND FUTURE WORK

## 9.1 Summary of Contributions

We presented semi-symbolic matching for aligning different types of information by converting visual image data into textual descriptions in visual and video question answering tasks. To exploit the extra textual information effectively, we proposed the three fusions for visual question answering and applied dual-attention and new losses for video question answering. Next, we introduced a novel instruction-following navigation and assembly dataset/task, called ARRAMON, in which agents are asked to follow natural language instruction to find a target object and place it to a designated location. We also suggested a new task setup from CVDN dataset, called NDH-FULL which addresses the issue of the existing NDH task which lacks full language supervision for finishing the task. Finally, we introduced new AI assistant systems: human body pose correctional guidance dataset/task, called FIXMYPOSE, and conversational image search and editing assistant, called CAISE. Models for both tasks should understand multimodal information (language+vision) to communicate with human users.

## 9.2 Future Work

**Beyond Factual Question Answering:** Visual question answering systems started as factual question answering. To answer the question *What is the mustache made of?*, a machine has to check what is the object hanging below someone's nose in the picture. However, for the systems to be more useful, they should reason beyond what is seen from an image/video (i.e., commonsense). Furthermore, the reasoning process should vary according to the condition change (i.e., counterfactual). For example, the system should be able to warn people by anticipating a negative implication when it is about to storm (e.g., by seeing the dark clouds coming in the sky). This reasoning process is not easy to learn for machines, but an important property to have. However, there are not enough datasets for building commonsense reasoning systems (especially counterfactual reasoning). Therefore, collecting the dataset and building robust models on the dataset are promising topics to pursue.

**Learning Language via Embodied Experience:** Due to the great success of deep learning recently, machines now can read and see in some context. However, for building integrated intelligence, learning only from text or/and image might not be enough. Agents should learn by interacting with the environment to truly understand real-world phenomena. For example, an agent cannot capture the concept of *Turn Right* from a text until it performs the action in the environment and learns that the action causes the view change (showing the scene on the right-hand side). Therefore, infusing the information learned from embodied experience into language is an essential direction to explore. I would like to work on the topic by using simulated environments in which embodied agents learn language by interacting with the environment.

**Multimodal Embodied AI Assistant:** Imagine that Alexa comes into the AR glass in the form of an embodied avatar. We can directly interact with it using a multimodal medium like voice and gesture. Many applications would become possible too with the technology. For example, indoor navigation might be an interesting application, if a person is new to a building and wants to go to someone's office, the embodied virtual assistant can guide them to the destination. However, to build such systems, there remain many challenges to overcome. For example, an additional dimension of difficulty would be introduced when handling referring and spatial expressions (due to the increased degrees of freedom users introduce). Therefore, addressing such difficulties can be an important research topic. Since this is a relatively unexplored area, I would like to start by defining new tasks and identifying challenges that AI systems should overcome.

# APPENDIX A: SUPPLEMENTARY MATERIAL

## A.1 Attention Visualization

As shown in Figure A.1, paragraph captions contain direct or indirect clues for answering questions.

**The upper left figure** This is the case that a sentence in the paragraph caption can give obvious clue for answering the given question. By looking at the sentence “a boy is standing on the beach”, this question can be answered correctly.

**The upper right figure** The sentence “two cows are grazing in a field ” gives the correct answer “2” directly.

**The bottom left figure** There is no direct clue like “he is playing tennis”, but the correct answer can be inferred by integrating the information from different sentences such as “the man is holding a tennis racket” and “a man is standing on a tennis court”.

**The bottom right figure** This case seems tricky, but the answer can be inferred by associating the blue sky with daytime.

## A.2 ARRAMON: Task and Metrics

As shown in Figure A.2, the score of rPOD is decreased according to the placement error (the Manhattan distance) exponentially. Thus, to score high in the rPOD metric, agents should place the target objects as close to the target place as possible.

## A.3 ARRAMON: Dataset

To support the ARRAMON task, we collected a dataset. Our dataset is based on a large dynamic outdoor environment from which diverse instructions with interesting linguistic properties are derived.

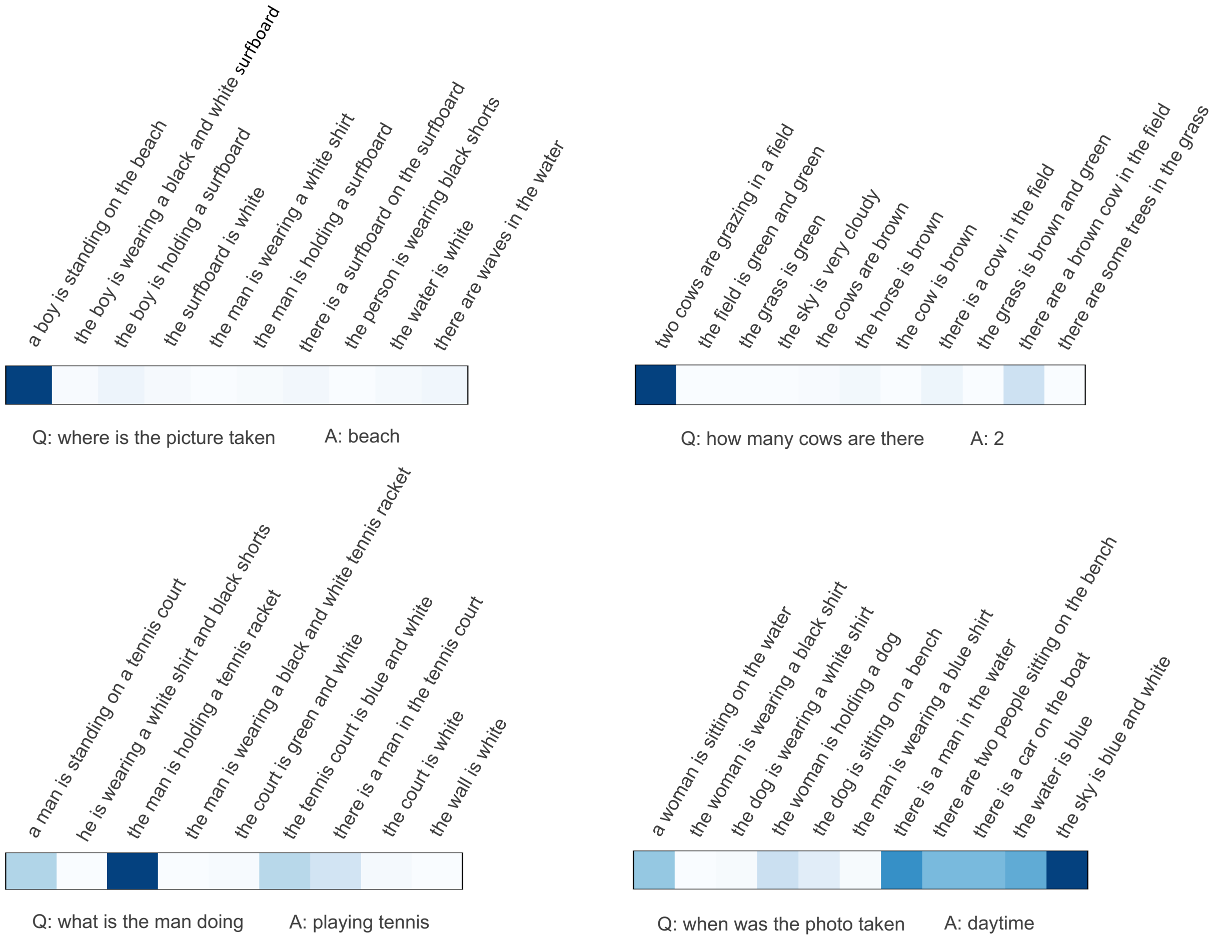
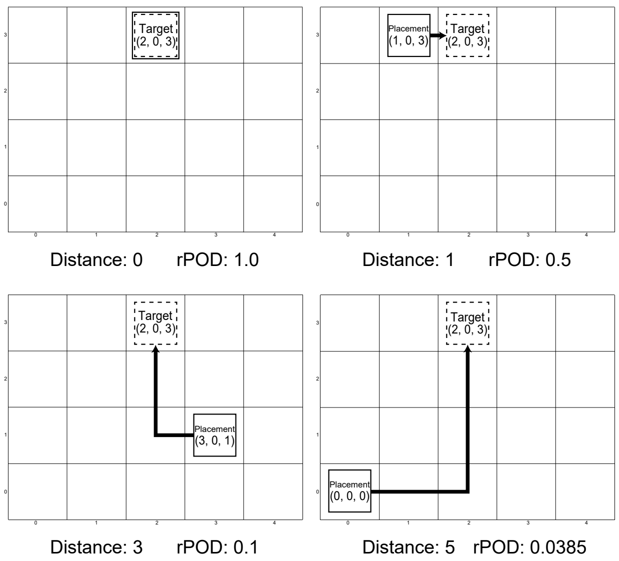


Figure A.1: Attention Visualization: For all examples, our model answers correctly.

### A.3.1 Data Collection

**Route Generation.** The ground truth trajectories is determined by the A* shortest path algorithm (Hart et al., 1968). Using the shortest path algorithm allows the resulting Ground Truth (GT) path to be straightforward and reach the target while avoiding going to unnecessary places. The blue navigation guideline provided to the Stage 1 workers is a mimic of this GT path (Figure A.8a).

**Qualification Tests.** When placing an object in the assembly phase, the item is placed 1 space in front of where the agent stands. To ensure that the workers who will be following instructions in Stage 2 fully understood this concept, at the start of Stage 2, they were presented with a small test (Figure A.3) that would show them how to correctly move and place objects and required that they demonstrate that they could do so. Both Stage 1 and 2 workers were also required to pass a



Distance: 0 rPOD: 1.0

Distance: 1 rPOD: 0.5

Distance: 3 rPOD: 0.1

Distance: 5 rPOD: 0.0385

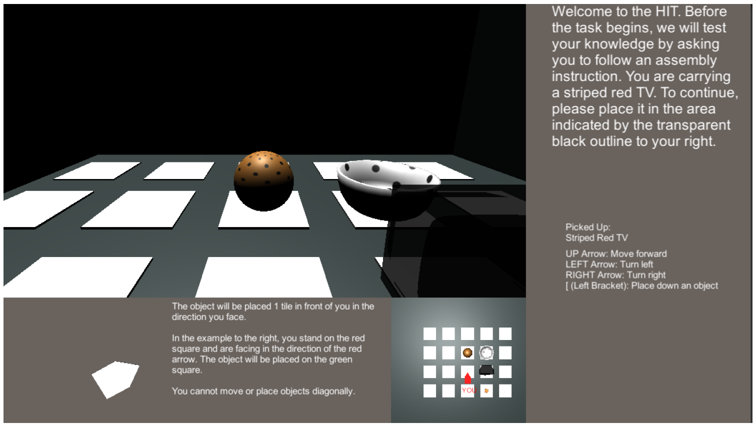
Figure A.2: Distance and rPOD metric: as the Manhattan distance between target and agent placement locations increases, the rPOD score decreases exponentially.

short screening test before they could begin their respective tasks. The tests are shown in Figure A.4 (Stage 1) and Figure A.5 (Stage 2).

**Worker Bonus Criteria and Rates.** For Stage 1 workers who did the instruction writing task correctly {5, 20, 50} times, a bonus of {\$0.10, \$0.90, \$4.00} respectively was awarded. Stage 1 workers were also provided a \$0.10 bonus for every instruction they wrote that was able to successfully pass Stage 2 verification with high nDTW and perfect assembling scores.

**Instruction Rules and Guidelines.** Rules and guidelines were put into place to help ensure that instructions written by the Stage 1 workers were high quality and written with as few errors as possible. Particularly, the guidelines serve to prevent the workers from using other elements of the UI or tools we provided, such as the blue navigation line or guiding arrow (see Figure A.8) and other elements that were not part of the true environment in their instructions.

- Instructions must be written relative to objects and the environment and not contain exact counts of movements (e.g., “Go forward 10 times and then turn left 2 times” is bad).
- Instructions must be clear, concise, and descriptive.
- Do not write more than the text-field can hold.



Welcome to the HIT. Before the task begins, we will test your knowledge by asking you to follow an assembly instruction. You are carrying a striped red TV. To continue, please place it in the area indicated by the transparent black outline to your right.

Picked Up:
Striped Red TV

UP Arrow: Move forward
LEFT Arrow: Turn left
RIGHT Arrow: Turn right
[ (Left Bracket): Place down an object

The object will be placed 1 tile in front of you in the direction you face.

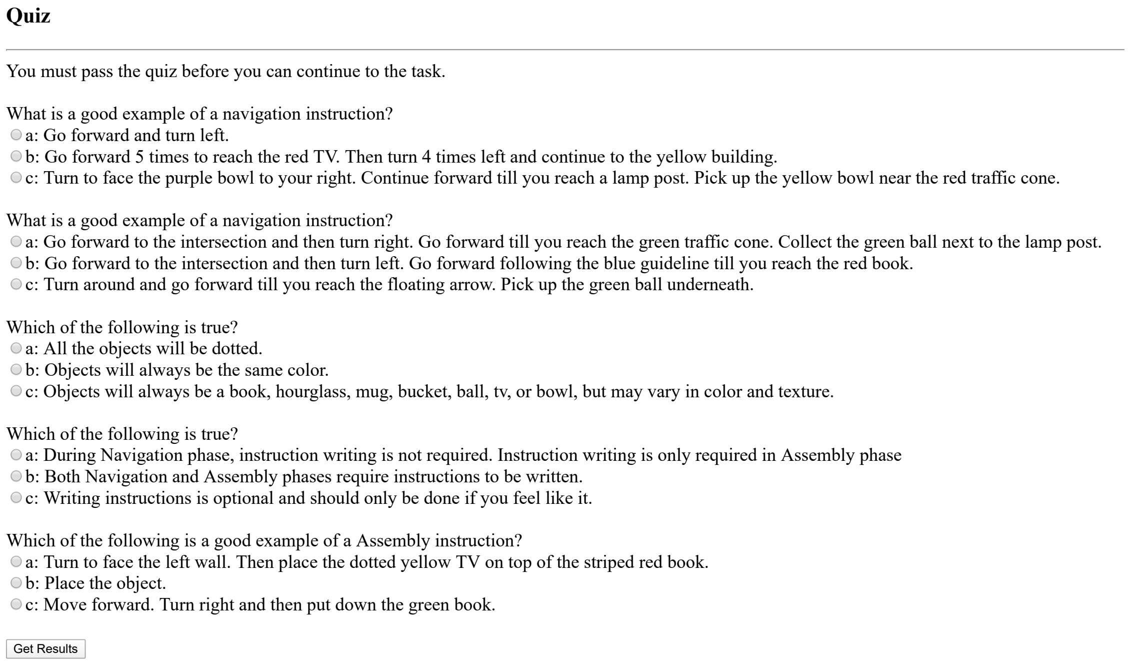
In the example to the right, you stand on the red square and are facing in the direction of the red arrow. The object will be placed on the green square.

You cannot move or place objects diagonally.

Figure A.3: Illustration of the assembly phase test before the start of stage 2.

- At the end of writing an instruction for the navigation phase be sure to include something similar to "pick up" or "collect" the object.
- At end of writing an instruction for the assembly phase be sure to include something similar to "place" or "put" the object you collected before.
- Do not reference the navigation line, the blue balls on the navigation line, the floating arrows above the objects, or any of the interface elements when writing instructions.
- Do not reference any buildings that are a solid gray color.
- Do not reference the transparent black outline or the white grid tiles on the floor (Figure A.7 and Figure A.8b) during the assembly phase
- Do not write vague or potentially misleading instructions and do not create any instructions that reference previous instructions such as "Go back to" or "Return to".
- Avoid spelling and grammar mistakes.
- When writing instructions for the assembly phase, do not write movement instructions. Make sure to use object references (e.g, "the red dotted ball").

During the navigation phase, the instruction writing worker cannot stray from the navigation line, ensuring that they collect the objects in the correct order. During the assembly phase, regardless of where the instruction worker places the collected object, it will move into the correct position (workers are not informed of this), ensuring that the objects are always in the correct formation for the next phase and future instructions do not become invalid. Additionally, we have



**Quiz**

You must pass the quiz before you can continue to the task.

What is a good example of a navigation instruction?
○ a: Go forward and turn left.
○ b: Go forward 5 times to reach the red TV. Then turn 4 times left and continue to the yellow building.
○ c: Turn to face the purple bowl to your right. Continue forward till you reach a lamp post. Pick up the yellow bowl near the red traffic cone.

What is a good example of a navigation instruction?
○ a: Go forward to the intersection and then turn right. Go forward till you reach the green traffic cone. Collect the green ball next to the lamp post.
○ b: Go forward to the intersection and then turn left. Go forward following the blue guideline till you reach the red book.
○ c: Turn around and go forward till you reach the floating arrow. Pick up the green ball underneath.

Which of the following is true?
○ a: All the objects will be dotted.
○ b: Objects will always be the same color.
○ c: Objects will always be a book, hourglass, mug, bucket, ball, tv, or bowl, but may vary in color and texture.

Which of the following is true?
○ a: During Navigation phase, instruction writing is not required. Instruction writing is only required in Assembly phase
○ b: Both Navigation and Assembly phases require instructions to be written.
○ c: Writing instructions is optional and should only be done if you feel like it.

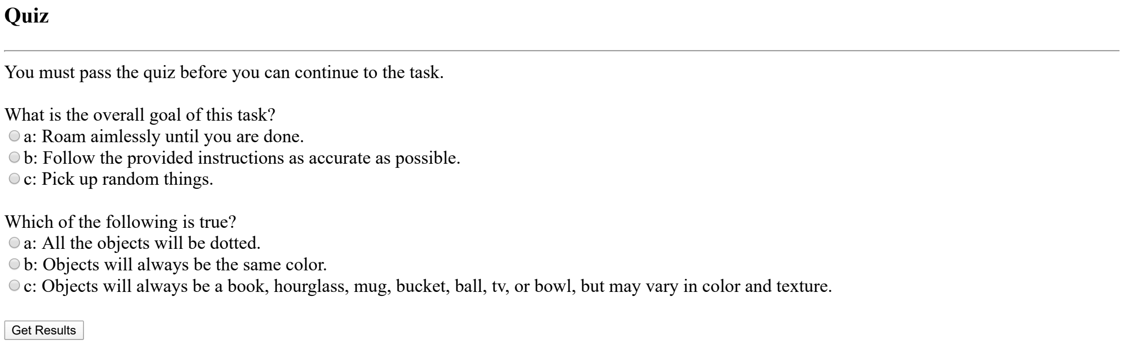
Which of the following is a good example of a Assembly instruction?
○ a: Turn to face the left wall. Then place the dotted yellow TV on top of the striped red book.
○ b: Place the object.
○ c: Move forward. Turn right and then put down the green book.

Figure A.4: Screening test that is required to be taken prior to starting Stage 1.

implemented active quality checks which will prevent a worker from submitting their instructions if certain criteria is not met. If a worker is blocked by one of these checks, they will be shown which check failed so that they can easily correct the error.

**General Active Quality Checks.**

- Each instruction must contain at least 6 words.
- Less than 40% of the characters in the instructions can be spaces.
- The symbols (, [, ], ), &, *, ^, %. $, #, @, !, =, and + cannot be included.
- Single letter words other than ”a” cannot be included.
- A single letter cannot be repeated 3 consecutive times. i.e “sss”.
- The same word cannot be repeated twice in a row.
- At least 40% of the words in the instruction must be unique.
- The term “key” cannot be included.
- The term “step” cannot be included.



**Quiz**

You must pass the quiz before you can continue to the task.

What is the overall goal of this task?
○ a: Roam aimlessly until you are done.
○ b: Follow the provided instructions as accurate as possible.
○ c: Pick up random things.

Which of the following is true?
○ a: All the objects will be dotted.
○ b: Objects will always be the same color.
○ c: Objects will always be a book, hourglass, mug, bucket, ball, tv, or bowl, but may vary in color and texture.

Figure A.5: Screening test that is required to be taken prior to starting Stage 2.

- The term “time” cannot be included.
- The term “go back” cannot be included.
- The term “return” cannot be included.
- The term “came” cannot be included.
- The term “item” cannot be included.

**Navigation Active Quality Checks.**

- If the ground truth path requires turning at the beginning of the path, the term “turn” must be included.
- The term “arrow” cannot be included.

**Assembly Active Quality Checks.**

- The terms “tile” or “grid” cannot be included.
- The term “space” cannot be included.
- The term “go” cannot be included.
- The term “corner” cannot be included.
- The term “move” cannot be included.
- The black outline cannot be referenced.

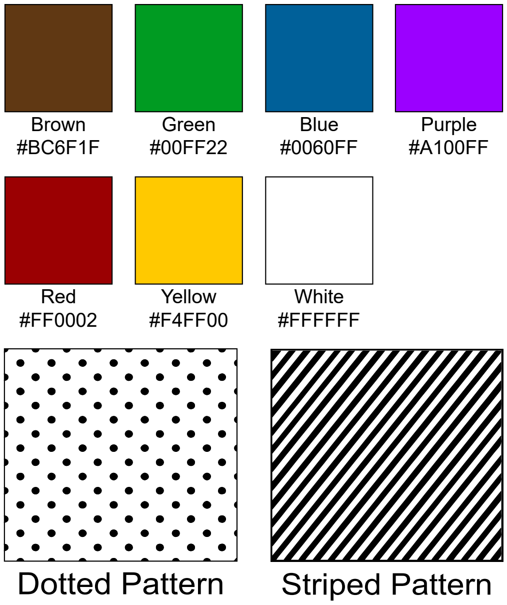


Figure A.6: Illustration of the colors and patterns that collectable and distracter objects can have.

**Review Notifications.** It is possible for instructions to be written that can pass all automated checks and still be of poor quality. However, there is no quick and reliable way to automatically check if an instruction passes the tests but is still vague or misleading. Additional active checks could be added, however, in cases of ambiguity, more active checks would result in potentially good instructions being blocked. Instead of blocking submission, checks that could have been incorrectly triggered, would send a notification email, allowing us to take quick action by manually reviewing the instruction in question to see if the worker who created it needs feedback on writing better instructions.

### A.3.2 Interface

**Stage 1: Instruction Writing.** The goal of this stage is to write instructions on how to navigate and place objects. The provided interface was designed to make this process easier for the workers completing the task. In both phases, the interface provides a arrow on the bottom left that will also point to the target destination and target location (depending on the active phase; navigation and assembly respectively.)

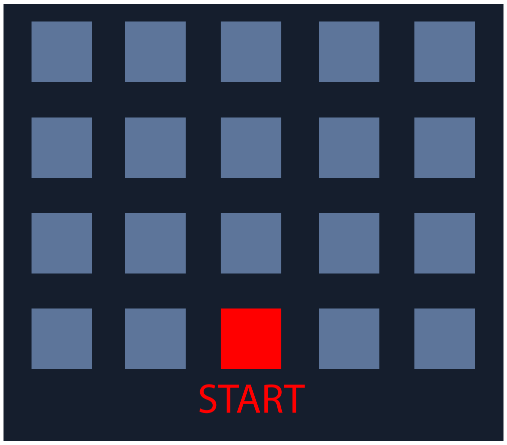
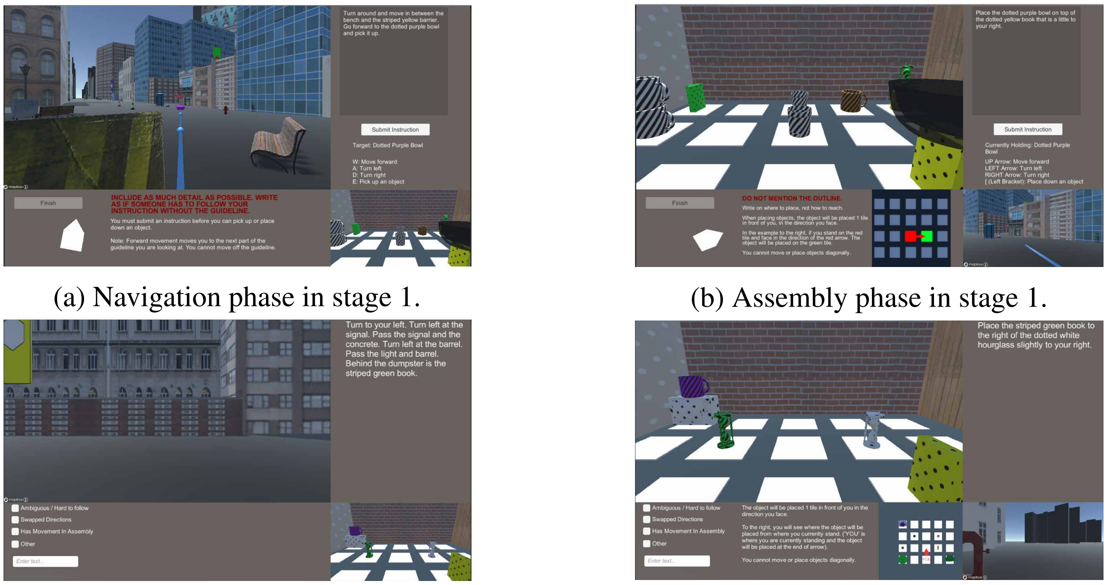


Figure A.7: Illustration of the assembly grid with the starting position marked.

- **Navigation Phase:** (Figure A.8a) The workers will follow the provided navigation line and as they follow it, write instructions on how to reach the destination. Additionally, the workers are provided with the controls and a few tips that they should keep in mind while completing the navigation phase. A small preview of the next phase (Assembly) is shown in the lower right.
- **Assembly Phase:** (Figure A.8b) The interface is similar to that of the navigation phase interface. During this phase, the Assembly preview which previously occupied the lower right corner will come into focus, and the navigation phase preview is now occupying that space. In this phase, no navigation line is provided, as there is nowhere that cannot be seen from the starting position. The controls and tip information are updated with information about the assembly phase.

**Stage 2: Instruction Following.** The goal of this stage is for the instructions written in the previous to be validated. Again, this interface was designed to make completing this task easier for the workers. Workers are also provided with some check boxes, which they can use to flag an instruction for certain issues so that we can more easily identify poor instructions.

- **Navigation Phase:** (Figure A.8c) Workers are placed in an exact copy of the environment that a Stage 1 worker used, as well as given the instructions they wrote on how to accomplish the task, which are visible in the top right corner. This new worker is not provided the blue guideline and the indicating arrow, and must now navigate using the instructions alone.



Turn around and move in between the bench and the striped yellow barrier. Go forward to the dotted purple bowl and pick it up.

Target: Dotted Purple Bowl

W: Move forward
A: Turn left
D: Turn right
E: Pick up an object

INCLUDE AS MUCH DETAIL AS POSSIBLE. WRITE AS IF SOMEONE HAS TO FOLLOW YOUR INSTRUCTION WITHOUT THE GUIDELINE.

You must submit an instruction before you can pick up or place down an object.

Note: Forward movement moves you to the next part of the guideline you are looking at. You cannot move off the guideline.

(a) Navigation phase in stage 1.

Place the dotted purple bowl on top of the dotted yellow book that is a little to your right.

Currently Holding: Dotted Purple Bowl

UP Arrow: Move forward
LEFT Arrow: Turn left
RIGHT Arrow: Turn right
[ (Left Bracket): Place down an object

DO NOT MENTION THE OUTLINE.

Write on where to place, not how to reach.

When placing objects, the object will be placed 1 tile in front of you, in the direction you face.

In the example to the right, if you stand on the red tile and face in the direction of the red arrow. The object will be placed on the green tile.

You cannot move or place objects diagonally.

(b) Assembly phase in stage 1.

Turn to your left. Turn left at the signal. Pass the signal and the concrete. Turn left at the barrel. Pass the light and barrel. Behind the dumpster is the striped green book.

☐ Ambiguous / Hard to follow
☐ Swapped Directions
☐ Has Movement In Assembly
☐ Other

Enter text...

(c) Navigation phase in stage 2.

Place the striped green book to the right of the dotted white hourglass slightly to your right.

☐ Ambiguous / Hard to follow
☐ Swapped Directions
☐ Has Movement In Assembly
☐ Other

Enter text...

The object will be placed 1 tile in front of you in the direction you face.

To the right, you will see where the object will be placed from where you currently stand. ('YOU' is where you are currently standing and the object will be placed at the end of arrow).

You cannot move or place objects diagonally.

(d) Assembly phase in stage 2.

Figure A.8: Simulation Interfaces of the Stage 1 (upper), Stage 2 (lower) showing separate examples, navigation phase (left), and assembly phase (right) of the data collection. (a) Workers are initially shown the navigation phase interface and must follow the blue navigation line to the target objects and write instructions as they go. (b) Workers are moved into assembly and must make assembly instructions guided by the highlighted (transparent black) objects. (c) Workers are provided with the navigation instructions and must find the target objects identified by the instructions. (d) Workers are provided with the assembly instructions and must place the collected object at the target position identified by the instructions.

- **Assembly Phase:** (Figure A.8d) The worker is again shifted into the assembly room, but will no longer see the transparent outline that indicates where the object should be placed. They must instead rely on the instructions written by a Stage 1 worker. The worker is also provided a real-time diagram indicating where they will place the object given the position they currently stand. The object is always placed 1 space directly in front of the worker's location. The worker is also provided with some tips that might help them.

## A.4 ARRAMON: Model

**Cross Attention.** We employ the bidirectional attention mechanism (Seo et al., 2017) to align the visual feature $V$ and instruction feature $L$. We calculate the similarity matrix, $S \in \mathbb{R}^{w' \times l}$ between visual and instruction.

$$S_{ij} = W_s^\top (V_i \odot L_j) \tag{A.1}$$

where $W_s \in \mathbb{R}^{d \times 1}$ is the trainable parameter, and $\odot$ is element-wise product. From the similarity matrix, the new fused instruction feature is:

$$\bar{V} = \text{softmax}(S^\top)V \tag{A.2}$$

$$\hat{L} = W_L^\top [L; \bar{V}; L \odot \bar{V}] \tag{A.3}$$

Similarly, the new fused visual feature is:

$$\bar{L} = \text{softmax}(S)L \tag{A.4}$$

$$\hat{V} = W_V^\top [V; \bar{L}; V \odot \bar{L}] \tag{A.5}$$

where $W_L$ and $W_V$ are trainable parameters.

**General Attention.** We employ a basic attention mechanism for aligning action feature, $h$, and each of visual and instruction features.

$$A_i = \hat{V}_i^\top h \tag{A.6}$$

$$\alpha = \text{softmax}(A) \tag{A.7}$$

$$v = \alpha^\top \hat{V} \tag{A.8}$$

## A.5 ARRAMON: Experiments

### A.5.1 Simulator Setup

Our task is quite challenging. In many cases, agents may not even be able to pick up an object in the navigation phase (agents would have to be in a position close enough to the object and of the correct rotation to pick the object. These factors along with the size of the environment, make

| Model | | Val Seen | | | | | | | Val Unseen | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Navigation | | | | | Assembly | | Navigation | | | | | Assembly | |
| | | nDTW | CTC | | | | rPOD | PTC | nDTW | CTC | | | | rPOD | PTC |
| | | | k=0 | k=3 | k=5 | k=7 | | | | k=0 | k=3 | k=5 | k=7 | | |
| V/L | T1 | 0.222 | 0.000 | 0.138 | 0.194 | 0.260 | 0.088 | 0.070 | 0.186 | 0.000 | 0.080 | 0.139 | 0.192 | 0.054 | 0.044 |
| | T2 | 0.049 | 0.000 | 0.057 | 0.103 | 0.140 | 0.027 | 0.017 | 0.033 | 0.000 | 0.044 | 0.078 | 0.113 | 0.019 | 0.011 |
| | total | 0.135 | 0.000 | 0.098 | 0.149 | 0.200 | 0.058 | 0.044 | 0.109 | 0.000 | 0.062 | 0.108 | 0.153 | 0.036 | 0.028 |

Table A.1: Performance of Vision-and-Language (V/L) baseline for turns T1 and T2, plus overall scores on the Val-Seen/Unseen splits.

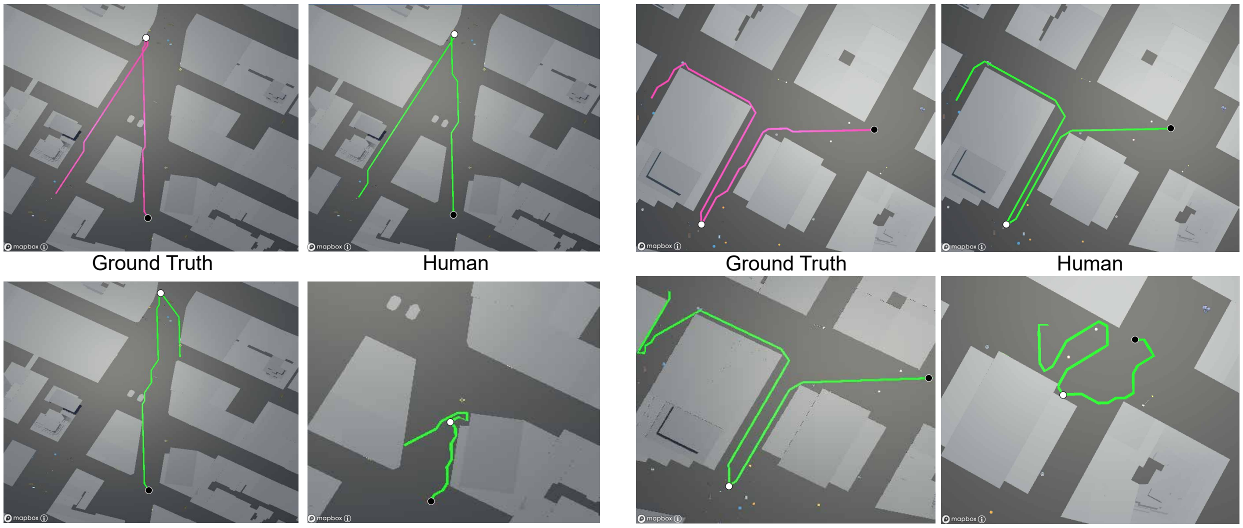
| Model | Navigation | Assembly | |
|---|---|---|---|
| | CTC (k=3) | rPOD | PTC |
| Vision-and-Language | 1.000 | 0.539 | 0.382 |

Table A.2: Scores in the assembly phase calculated under the assumption of the perfect performance in the navigation phase on Val-Seen split.

this difficult). To decrease the difficulty of the task, in the event agents do not successfully pick up an object, we allow them to continue to the assembly phase with whatever object is the closest to their final location. Likewise in the assembly phase, if the time step limit is reached before the agent places the object down, the object will be placed in front of them (in the event "in front of them" is out of bounds, it is placed at their feet). Note that either of these actions will result in PTC and rPOD to be 0.

### A.5.2 Training Details

We use PyTorch (Paszke et al., 2017) to build our model. We take the average of the losses from navigation and assembly phase modules to calculate the final loss. We use 128 as a hidden size of linear layers and LSTM. For word and action embedding sizes, we use 300 and 64, respectively. The visual feature map size is 7 × 7 with 2048 channel size. For dropout p value, 0.3 is used. We use Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to 0.001. The number of trainable parameters of our Vision-and-Language model is 1.83M (Language-only: 1.11M, Vision-only: 0.73M). We use NVIDIA RTX 2080 Ti and TITAN Xp for training and evaluation, respectively.



Ground Truth | Human | Ground Truth | Human

Our Model | Random Walk | Our Model | Random Walk

**Turn 1 ● :** Turn slightly left as you move ahead past the traffic light. Go toward the speed limit sign, and move past the dotted white barrier. Head to the left to the lamp post, and fetch the dotted brown tv past a blue cone.

**Turn 2 ○ :** Turn around and pass the blue and orange cones. Keep going straight for a long way passing the speed limit sign. Head toward the two striped yellow barriers ahead, but pick up the striped yellow book before you reach them.

**Turn 1 ● :** Turn right until you see the green banner. Go towards the tire stack to the right of it and take a left down the street behind it. Go forward and pass the barrel. In the intersection there is a dotted white bucket. Pick up the dotted white bucket.

**Turn 2 ○ :** Turn right until you see the green cone. Go forward and take a left at the first street. Go towards the trash bags and take a left at the street. Pass the black barrel and go towards the dotted blue bucket. Pick up the dotted blue bucket.

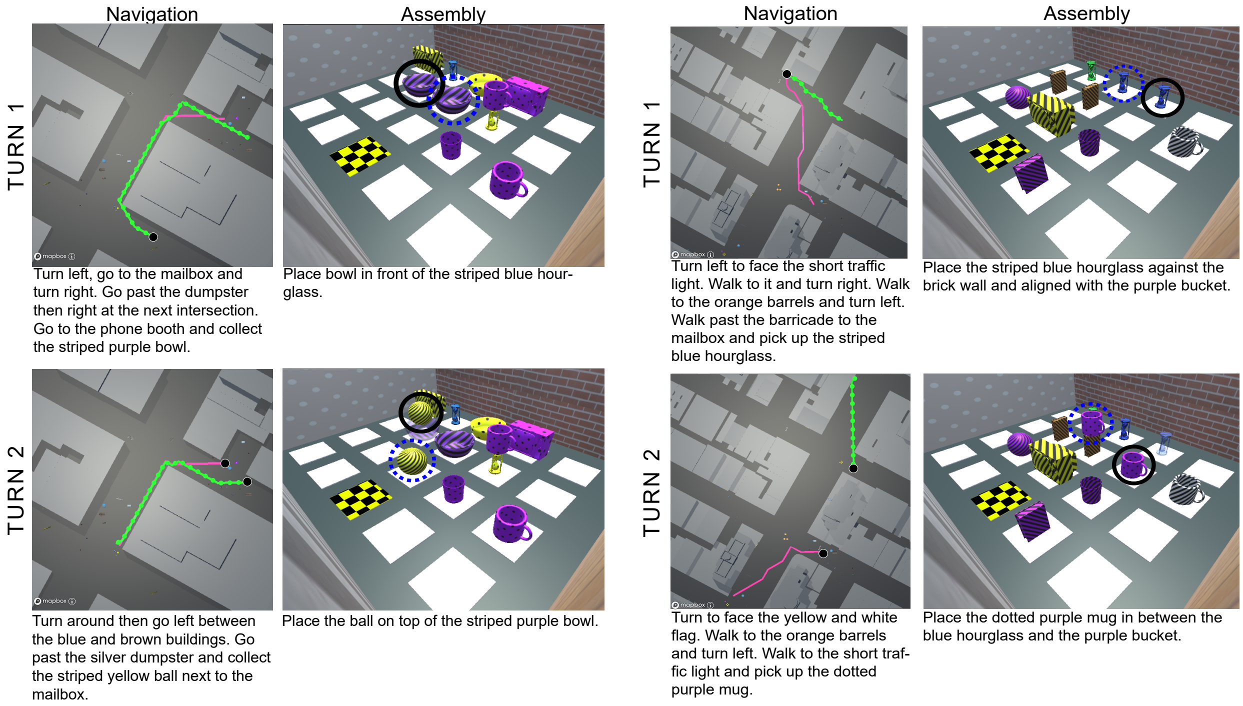
Figure A.9: Navigation paths of ground truth, human evaluation, random walk, and our model. Pink is the GT path and the other paths are shown in green (turn 1 starts from the black dot and goes to the white dot. Turn 2 starts from white dot and goes to the end of the path).

## A.6 ARRAMON: Results and Analysis

As shown in Table A.1, almost all scores from turn 1 are improved compared to turn 2. Scoring in rPOD and PTC metrics in the assembly phase is largely dependent on the score of CTC-k in the navigation phase. Comparing the rPOD and PTC scores of Vision-and-Language model on the val-seen split (Table A.1) and the ones from Table A.2, if the CTC-k is decreased by 1/10 (1.0 to 0.098), the PTC is also decreased around 1/10 (0.382 to 0.044). This demonstrates our ARRAMON task involves interweaving and is challenging to complete.

## A.7 ARRAMON: Output Examples

In the left path set of Figure A.9, our model follows the instructions well in the beginning. However, the model goes a little bit further and fails to find the target object (dotted brown tv).



Turn left, go to the mailbox and turn right. Go past the dumpster then right at the next intersection. Go to the phone booth and collect the striped purple bowl.

Place bowl in front of the striped blue hourglass.

Turn left to face the short traffic light. Walk to it and turn right. Walk to the orange barrels and turn left. Walk past the barricade to the mailbox and pick up the striped blue hourglass.

Place the striped blue hourglass against the brick wall and aligned with the purple bucket.

Turn around then go left between the blue and brown buildings. Go past the silver dumpster and collect the striped yellow ball next to the mailbox.

Place the ball on top of the striped purple bowl.

Turn to face the yellow and white flag. Walk to the orange barrels and turn left. Walk to the short traffic light and pick up the dotted purple mug.

Place the dotted purple mug in between the blue hourglass and the purple bucket.

Figure A.10: Visual demonstrations by our model in navigation and assembly phases. GT navigation paths are solid pink lines and model's paths are dotted green lines (start = black dot). GT assembly target location is solid black circle and model's target object placement is dashed blue circle (start = checkered yellow tile, agent facing brick wall).

In the second turn, the model turns around, but does not do it fully, so heads a different direction failing to reach the goal position.

For the example on the right, the model performs very well in the first turn, but in the second turn fails to find the target object although reaches very close to it and then backtracks out of the alley. Also, as shown in the figure, the human performs the navigation almost perfectly, indicating there is significant room for improvement by future work, and random-walk shows quite poor performance, implying that our ARRAMON task cannot be completed by random chance.

Figure A.10 compares the model against the GT in both turns and phases. On the left set, the model almost reaches the target object, but it cannot find the target object (striped purple bowl) and goes a little further past it. In the corresponding assembly phase, the model places the collected object (assuming it picked up the correct object in the previous navigation phase) 1 space to the right of the target location. In the next navigation turn, due to the error in the previous turn, the model path starts a bit further away from the GT, however, it starts to realign itself towards the end around the corner. The model is able to locate the target object and stop to pick it up. In the next assembly phase, the model fails to place the collected object at the right location. On the right set, the model shows worse performance. It misses all of the turning needed to reach the target. In the assembly phase, the model misses the target location by 1 space, likely due to misunderstanding the complex spatial relationship in the instructions. In the next navigation phase, the model starts in the wrong place, so ends up arriving at a totally different place from the target position. In the next assembly phase, the performance of the previous turn affected the object configuration, so the model cannot find the place "between the blue hourglass and the purple bucket".

### A.8 FIXMYPOSE: Distractor Choice Criteria

For the "target" and distractor images of the target-pose-retrieval task, we only consider images that meet these criteria: (1) the "target" pose image must have more than 10cm average joints distance from the "current" pose image, (2) each distractor has an average joints distance between 10cm and 1m from the "target" pose image, (3) each distractor must have less than 2m average joints distance from the "current" pose image, (4) the "current" pose image is included in the distractor images, and (5) all the distractor images are from the same environment and have the same character as the one in the "target" pose image.

### A.9 FIXMYPOSE: Dataset

#### A.9.1 Image and Environment Generation

**Environment Creation.** Every object inside of each room is collected from free assets in the Unity Asset Store[1] and various other free online resources. The first room is also collected as a free asset from the Unity Asset Store, however the rest of the rooms are manually created. In

[1] https://assetstore.unity.com

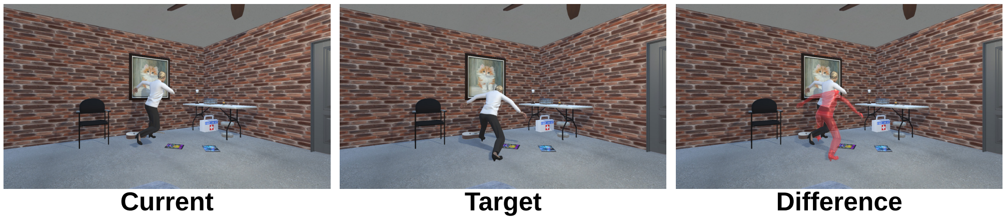


Figure A.11: Current, target, and difference images. The target images are taken 10 frames after the current images are taken. The difference image shows the overlap of the "current" and "target" images with the pose in the "target" image shown in red.

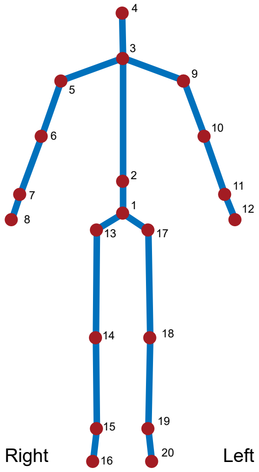
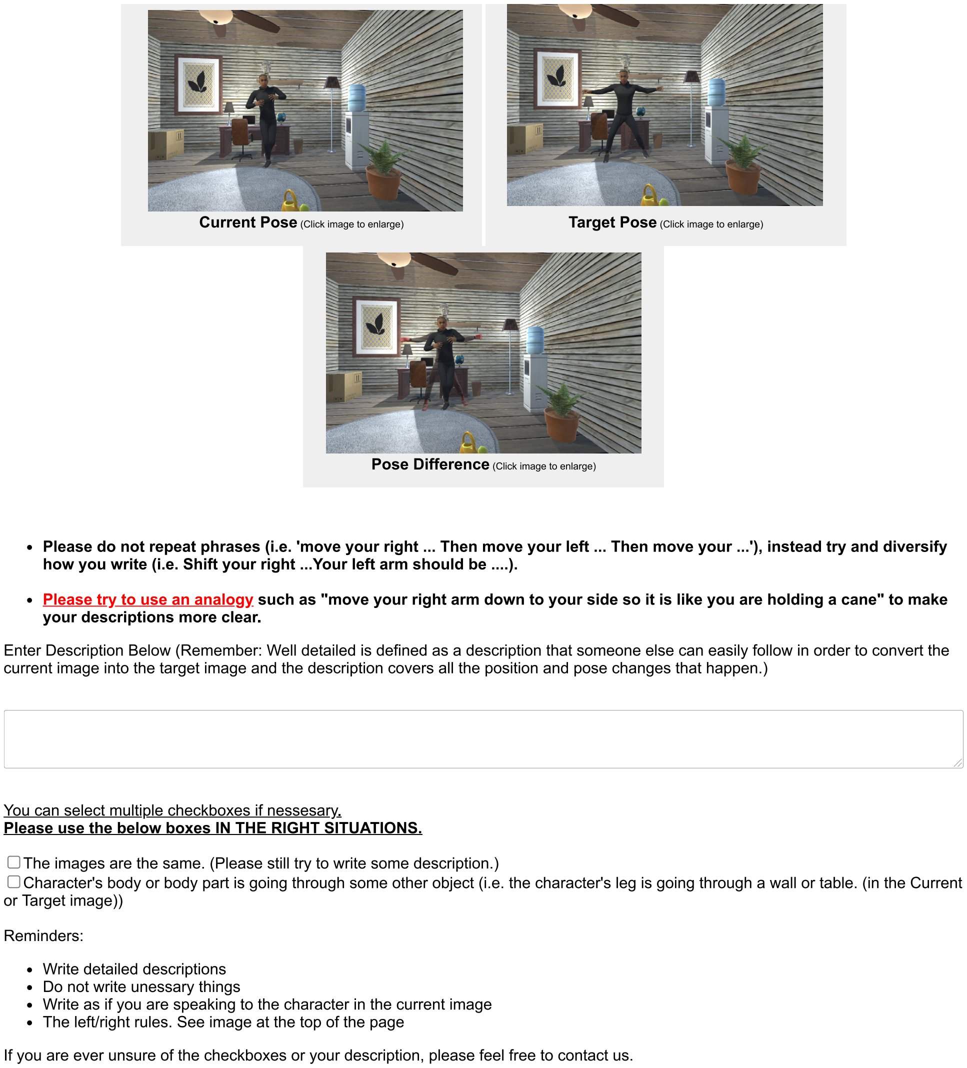


Figure A.12: The 3D joint configuration of characters (from index 1 to 20: center hip, spine, neck, head, right shoulder/elbow/wrist/hand, left shoulder/elbow/wrist/hand, right hip/knee/ankle/foot, left hip/knee/ankle/foot).

all rooms, including the first room, we manually choose and configure the arrangement of the objects.

**Image Capture.** To obtain each pair of images, we run the same animation twice but the second instance of the animation is offset by 10 animation frames. The 10 frame offset helps ensure that a clear visual difference is created, but not so much that it creates two unrelated images. The image of the first instance is the "current" image and the image of the second instance is the "target" image. Fig. A.11 shows an example of a "current" and "target" image as well as a "difference" image which shows the overlap of the "current" and "target" images with the pose in the "target" image shown in red. Every 20 frames, an image pair is captured.



Current Pose (Click image to enlarge)

Target Pose (Click image to enlarge)

Pose Difference (Click image to enlarge)

- **Please do not repeat phrases (i.e. 'move your right ... Then move your left ... Then move your ...'), instead try and diversify how you write (i.e. Shift your right ...Your left arm should be ....).**
- **Please try to use an analogy such as "move your right arm down to your side so it is like you are holding a cane" to make your descriptions more clear.**

Enter Description Below (Remember: Well detailed is defined as a description that someone else can easily follow in order to convert the current image into the target image and the description covers all the position and pose changes that happen.)

You can select multiple checkboxes if nessesary.
**Please use the below boxes IN THE RIGHT SITUATIONS.**

☐ The images are the same. (Please still try to write some description.)
☐ Character's body or body part is going through some other object (i.e. the character's leg is going through a wall or table. (in the Current or Target image))

Reminders:

- Write detailed descriptions
- Do not write unessary things
- Write as if you are speaking to the character in the current image
- The left/right rules. See image at the top of the page

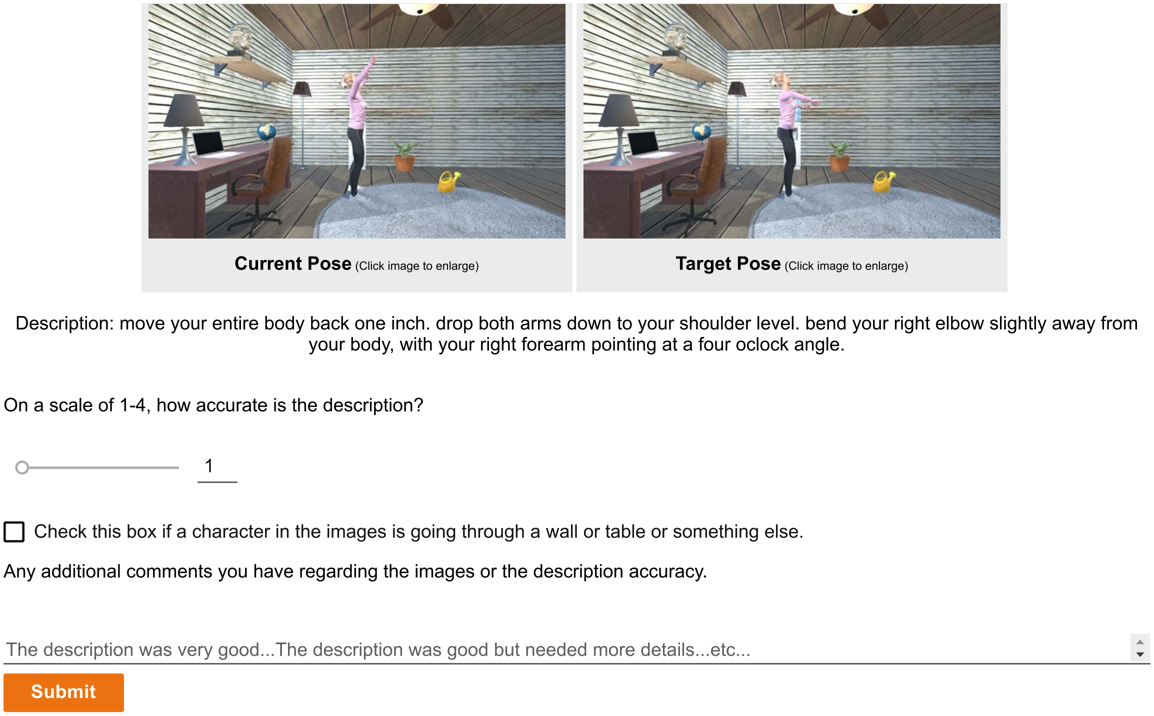
If you are ever unsure of the checkboxes or your description, please feel free to contact us.

Figure A.13: The interface of the writing task.

**3D Body Joint Data.** We obtain the 3D positional joint data of the character's poses from both the "current" and "target" images (see Fig. A.12). The positional data is relative to the camera's position and angle. This keeps all the data normalized regardless of which room or location in a room is chosen.

### A.9.2 Data Collection Interface

For each of the 3 data collection tasks (writing, verification, translation), we create a separate interface. The writing task and verification task are also provided with certain flags (detailed



**Current Pose** (Click image to enlarge)

**Target Pose** (Click image to enlarge)

Description: move your entire body back one inch. drop both arms down to your shoulder level. bend your right elbow slightly away from your body, with your right forearm pointing at a four oclock angle.

On a scale of 1-4, how accurate is the description?

○——— 1

□ Check this box if a character in the images is going through a wall or table or something else.

Any additional comments you have regarding the images or the description accuracy.

The description was very good...The description was good but needed more details...etc...

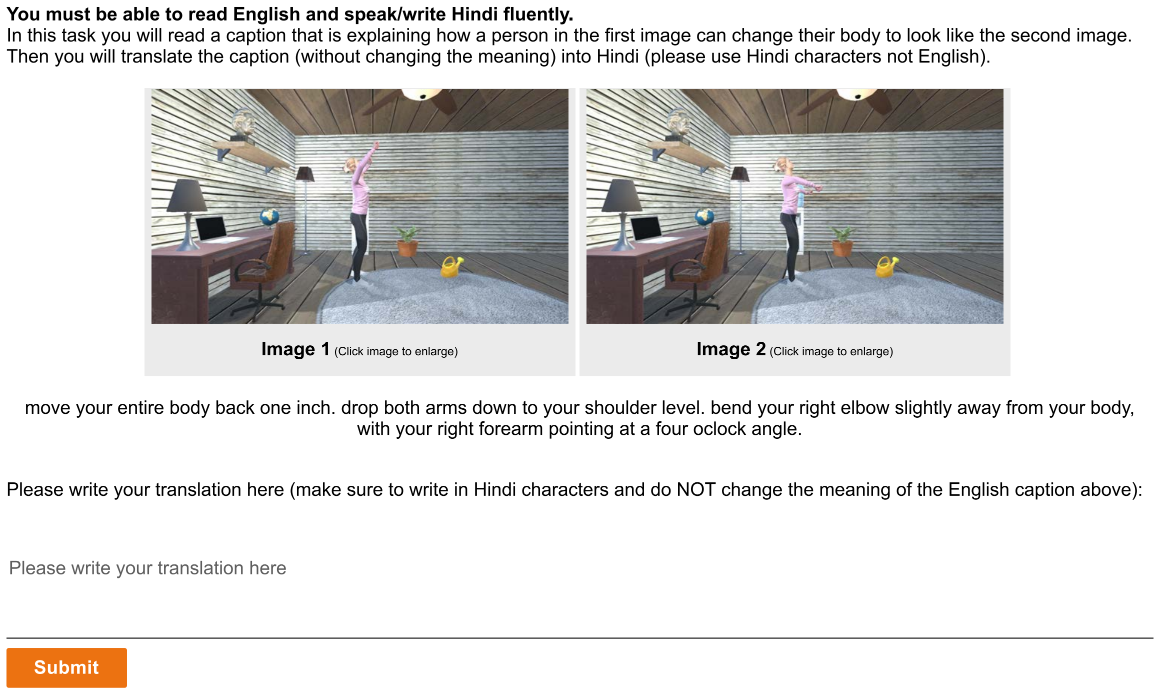
Submit

Figure A.14: The interface of the verification task.

in corresponding interface paragraphs). Upon clicking the images in any of the interfaces, the clicked image will be enlarged and an option to view the image in a separate tab is given in case the worker would like an even larger image.

**Writing Task Interface.** During this task, the goal is to have the workers see the 3 images ("current", "target", "difference") and then write a correctional description based on those images. The interface (as shown in Fig. A.13) provides the 3 images labeled and a writing area. Workers are also provided with "no clear difference" and "character is going through an object" flag. The "no clear difference" flag is designed to be used in the case the difference between the poses in the "current" image and "target" image is too small to write a good description. The "character is going through an object" flag is meant to be used in the event that a character in either image has a body part going through a wall, table, or any other object.

**Verification Task Interface.** This task serves to filter out any descriptions that are of poor quality. To do this, workers are provided with the "current" image and the "target" image and



**You must be able to read English and speak/write Hindi fluently.**
In this task you will read a caption that is explaining how a person in the first image can change their body to look like the second image.
Then you will translate the caption (without changing the meaning) into Hindi (please use Hindi characters not English).

move your entire body back one inch. drop both arms down to your shoulder level. bend your right elbow slightly away from your body, with your right forearm pointing at a four oclock angle.

Please write your translation here (make sure to write in Hindi characters and do NOT change the meaning of the English caption above):

Image 1 (Click image to enlarge)
Image 2 (Click image to enlarge)
Please write your translation here
Submit

Figure A.15: The interface of the translation task.

then the correctional description that is written for that image pair. They are then asked to rank the quality of the description from 1-4, with 1 being the description is completely unrelated and 4 being the description is perfect. Then, just as for the writing task, a checkbox for the “character is going through an object” flag is provided in case the writing task workers miss it. The interface is shown in Fig. A.14.

**Translation Task Interface.** During this task, workers are asked to translate descriptions from English into Hindi. As shown in Fig. A.15, the interface provides the “current” and “target” images for context and then the English description. Then, a text field is provided where workers can write the translation.

### A.9.3 Data Collection Filters

During the writing task, some active quality checks are put in place to ensure that descriptions are of a certain base quality before they reach the verification task. Below is the list of each active quality that is put in place.

- Each description must contain at least 30 words.
- The symbols (, [, ], ), &, *, ^, %. $, #, and @ cannot be included.
- At least 50% of the words in the instruction must be unique.
- The term “image” cannot be included.
- The term “i” cannot be included.
- The term “target” cannot be included.
- The term “difference” cannot be included.

In the case that workers in writing task select the “no clear difference” checkbox on the interface, the 30-word minimum check is removed so that workers could write shorter descriptions, since there is not much difference to write about if the images are almost the same.

### A.9.4 Worker Qualifications and Incentives

There were a total of 356, 373, and 47 unique crowd-workers who successfully passed the qualifications and completed the writing, verification, and translation tasks, respectively, at least once.[2]

**Worker Qualifications.** For all 3 tasks, crowd-workers are required to pass certain qualifications before they could begin. As both writing and verification tasks require reading (and writing in the case of the writing task) English, we require workers to be from native-speaking English countries and as the translation task requires translating to Hindi, we require that workers be from India. Crowd-workers are also required to have at least 1000 approvals from other tasks and a 95% or higher approval rating.

**Worker Payment.** The writing task takes around 1 minute and workers are paid $0.18 per description. For the first 25 high-quality descriptions that a worker writes, an additional bonus of $0.02 is given for each description and then for every subsequent 50 high-quality descriptions written, the bonus per description is increased by $0.01 ($0.02 bonus per description for first 25,

[2] We do not collect or use any private information from the workers.

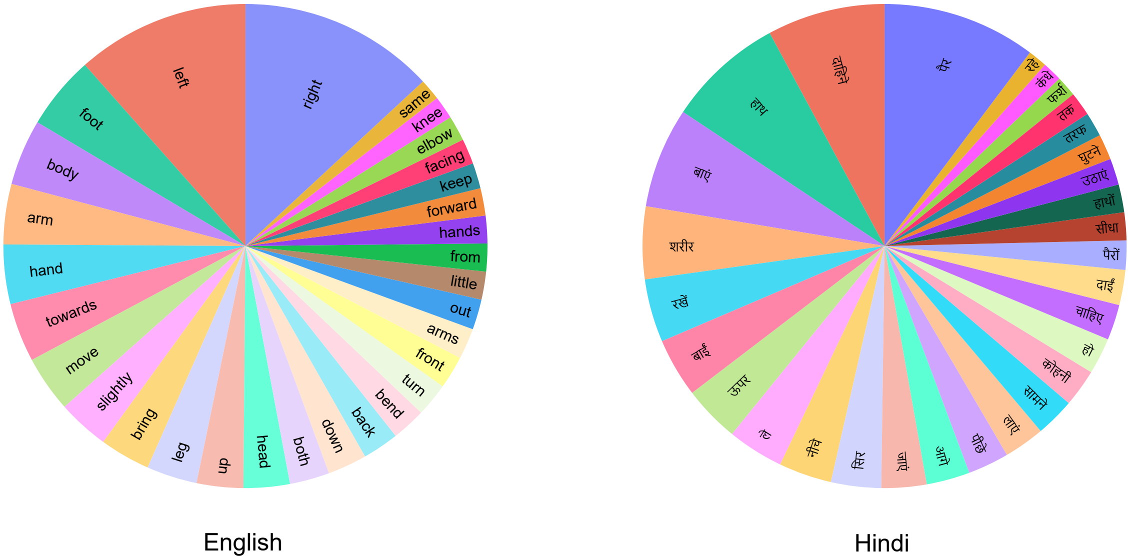


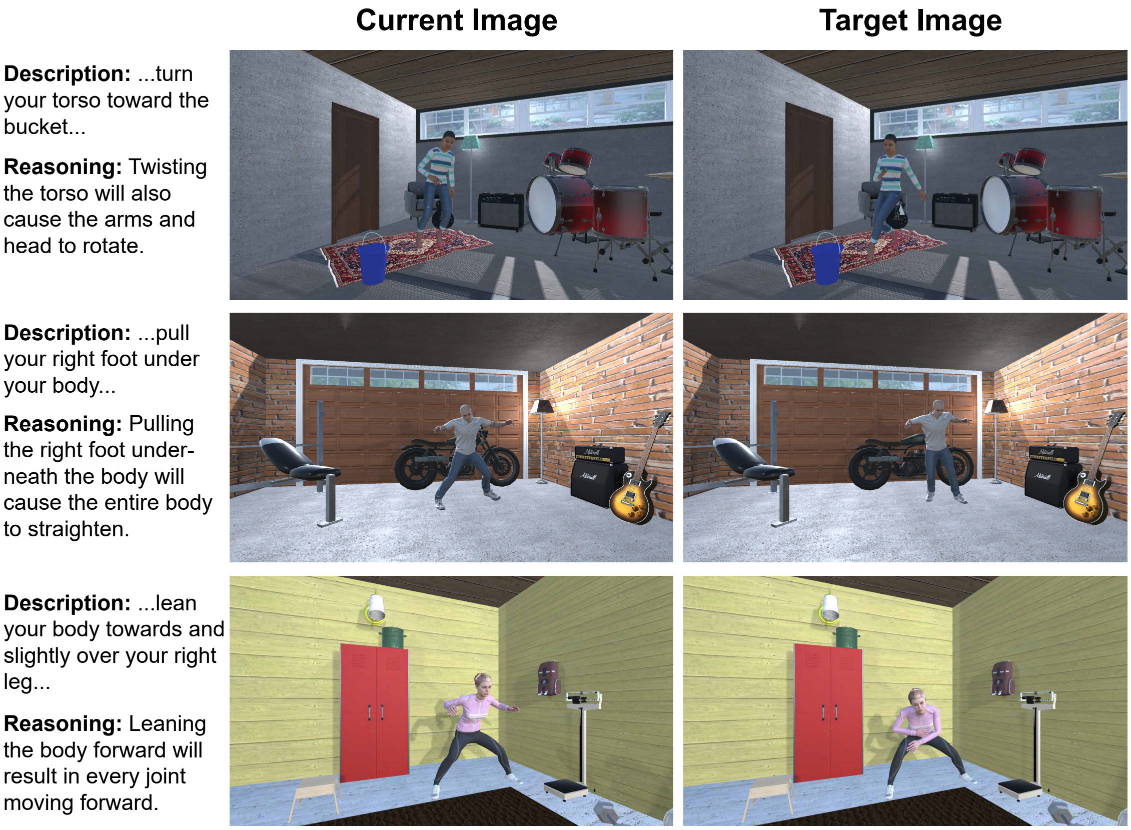
Figure A.16: The 30 most common English/Hindi words in the dataset (excluding stop words). They primarily relate to directions, body parts, and movements.

$0.03 bonus for the next 50, $0.04 bonus for the next 50, and so on). With this bonus rate, workers could get more than $0.20 quite easily since the task is not long (and hence overall reasonably higher than minimum hourly wages). Since there is no limit on how much a worker can write, they could potentially keep stacking the bonus as much as they want.

## A.10 FixMyPose: Analysis

### A.10.1 Most Commonly Occurring Words

The most commonly occurring words in our dataset are about direction, body parts, and movement, showing that models need to have a sense of direction with respect to body parts and objects, and also capture the differences between the poses to infer the proper movements. Fig. A.16 shows the most commonly occurring English/Hindi words in our dataset, which also primarily relate to directions, body parts, and movements.



**Description:** ...turn your torso toward the bucket...

**Reasoning:** Twisting the torso will also cause the arms and head to rotate.

**Description:** ...pull your right foot under your body...

**Reasoning:** Pulling the right foot underneath the body will cause the entire body to straighten.

**Description:** ...lean your body towards and slightly over your right leg...

**Reasoning:** Leaning the body forward will result in every joint moving forward.

Figure A.17: Examples of the ‘implicit movement description’ linguistic property.

### A.10.2 Description Length

The average length of the multi-sentenced descriptions (49.25 English / 52.74 Hindi words) is quite high, indicating that they are well detailed. The stddev (17.28/18.88) and the gap between the min and max (20/14 vs. 188/239) is quite large, reflecting the varying degrees of difference between the poses in an image pair. This length characteristic of the FIXMYPOSE dataset requires models to generate descriptions without being redundant or insufficient in detail.

### A.10.3 Linguistic Properties

The descriptions in the FIXMYPOSE dataset contain diverse linguistic properties. These properties as well as a few additional examples are provided in Table A.3. Additional examples of implicit movement description along with basic explanations are shown in Fig. A.17.

| Reference Frame | Freq. | Examples (English) |
|---|---|---|
| Egocentric Relation | 100% | “... **rotate your left shoulder** so that your hand is **above your elbow** ...”<br>“... put **your right foot and leg** forward so it is parallel **with your torso** ...”<br>“... move **your left leg** down and put in front of **your right leg** ...” |
| Environmental Direction | 52% | “... turn your left leg and right leg to the left to **face the wall with the door** ...”<br>“...turn your head to **look to the bed**...”<br>“...somewhat **aligning your eyes with the closest lamp** ...” |
| Implicit Movement Description | 58% | “... lean your body towards and slightly over your right leg ...”<br>“... rotate your torso slightly to the left ...”<br>“... then slightly lean forward ...” |
| Analogous Reference | 18% | “... extend your right arm straight in front of you **as if you are gesturing for someone to stop** ...”<br>“... twist your upper body back to your right **in a golf swing motion** ...”<br>“... hold your right hand next to your body **as if you are leaning on a cane** ...” |

Table A.3: Frequencies and detailed examples of the different properties present in correctional descriptions.

### A.11 FixMyPose: Models

**Cross Attention Stack.** CA-Stack is a stack of cross attentions.

$$\text{CA-Stack}(f^c, \hat{J}^c, f^t, \hat{J}^t) : \begin{cases} \bar{f}^c, \bar{J}^c = \text{CA}(f^c, \hat{J}^c) \\ \bar{f}^t, \bar{J}^t = \text{CA}(f^t, \hat{J}^t) \\ \tilde{f}^c, \tilde{J}^t = \text{CA}(\bar{f}^c, \bar{J}^t) \\ \tilde{f}^t, \tilde{J}^c = \text{CA}(\bar{f}^t, \bar{J}^c) \end{cases} \tag{A.9}$$

where CA is cross attention.

**Cross Attention.** We calculate the similarity matrix, $S$, between two features.

$$S_{ij} = f_i^\top g_j \tag{A.10}$$

From the similarity matrix, the new fused instruction feature is:

$$\hat{f} = \text{softmax}(S^\top) \cdot f \tag{A.11}$$

$$\bar{g} = W_g^\top [g; \hat{f}; g \odot \hat{f}] \tag{A.12}$$

Similarly, the new fused visual feature is:

$$\hat{g} = \text{softmax}(S) \cdot g \tag{A.13}$$

$$\bar{f} = W_f^\top [f; \hat{g}; f \odot \hat{g}] \tag{A.14}$$

where $W_g$ and $W_f$ are trainable parameters, $\odot$ is the element-wise product, and $\cdot$ is matrix multiplication.

**General Attention.** We employ a basic attention mechanism for aligning description features, $h$, and each of the visual and joints features.

$$A_i = f_i^\top h \tag{A.15}$$

$$\alpha = \text{softmax}(A) \tag{A.16}$$

$$\hat{f} = \alpha^\top f \tag{A.17}$$

**Self Gate.** We employ a basic attention mechanism for weighted summation of features.

$$A_i = \text{Linear}(k_i) \tag{A.18}$$

$$\alpha = \text{softmax}(A) \tag{A.19}$$

$$\hat{k} = \alpha^\top k \tag{A.20}$$

## A.12 FIXMYPOSE: Experiments

### A.12.1 Data Splits

For the pose-correction-captioning task, we split the dataset into train/val-seen/val-unseen/test-unseen. Since each room in our FIXMYPOSE has different visual setting (i.e., wall, floor, furniture, etc.), we assign separate rooms to val-unseen and test-unseen split. To be specific, we assign room 1 to 19, 24, and 25 to the train and val-seen splits, room 20 and 21 to the val-unseen, and room 22 and 23 to the test-unseen split. The final number of task instances for each split is 5,973/562/563/593 (train/val-seen/val-unseen/test-unseen) and the number of descriptions is 5,973/1,686/1,689/1,779. For the target-pose-retrieval task, we split the dataset into train/val-unseen/test-unseen. However, “unseen” in this task means “unseen animations”. The reason we split the dataset by animations is that, otherwise, the task would be easier by memorizing/capturing some patterns in the image pairs from certain animations. We assign animation 6 and 16 to val-unseen, 7 to test-unseen, and the rest of the animations to the train split. After filtering for the target candidates, we obtain 4,227/1,184/1,369 (train/val-unseen/test-unseen) instances.

### A.12.2 Human Evaluation Setup

We conduct human evaluation for the pose-correction-captioning task’s models to compare the output of the V-only (V: vision+joints) model, the L-only (L: language) model, and the full V+L model qualitatively. We randomly sample 100 generated descriptions from each model (val-seen split), then asked 3 random crowd-workers (we also applied the standard quality filters of above 95% hit success, over 1000 Hits, workers from native language-speaking countries) for each description to vote for the most relevant description in terms of the image pair, and for the one best in fluency/grammar (or ‘tied’).

Separately, to set the performance upper limit and to verify the effectiveness of our distractor choices for the target-pose-retrieval task, we conduct human evaluation. We randomly sample 50 instances from the target-pose-retrieval test-unseen split and ask an expert for the English and Hindi samples to perform the task. Human evaluation is conducted the same way for both English and Hindi. We also ask the expert to complete the task from a unimodal perspective (i.e., only given the “current” image or only given the description) to also show that the distractor choices cannot be exploited by any unimodal biases.

### A.12.3 Training Details (Reproducibility)

All of the experiments are run on a Ubuntu 16.04 system using the NVIDIA GeForce GTX 1080 Ti GPU and Intel Xeon CPU E5-2630. We employ PyTorch1.3 (Paszke et al., 2017) to build our models (torchvision0.4/Python3.5/numpy1.18/scipy1.4). The number of trainable parameters of the pose-correction-captioning V+L models are 9.1M and 9.6M for English and Hindi version, respectively (V-only: 10.4M/10.9M, L-only: 2.8M/3.1M), and the number of trainable parameters of the target-pose-retrieval V+L models are 10.9M/10.9M (V-only: 4.5M, L-only: 6.7M/6.7M). For the pose-correction-captioning task experiments, we use 9595/5555/2020 as the seed values, and run models 500 epochs and choose the best ones on val-seen/val-unseen splits. For the target-pose-retrieval task experiments, we use 5555/5556/5557 as the seed values, and run models 50 epochs and choose the best ones on the val-unseen split. In the pose-correction-captioning task model training, at training time, the models are trained with teacher-forcing approach, and at test time, the greedy-search is employed to generate the descriptions. For the multilingual model, we freeze the shared parameters at the point at which the English score is the highest, and then fine-tune specific non-shared modules for each language with ML and RL training. We employ ResNet-101 for the visual features. We use 512 as the hidden size and 256 as the word embedding dimension for both task models. We use the visual feature map of 7 × 7 with 2048 channel size for the pose-correction-captioning task models and $14 \times 14$ with 1024 channel size for the target-pose-retrieval task models. We use Adam (Kingma and Ba, 2015) as the optimizer and set the learning rate to $1 \times 10^{-4}$ for ML training (for both tasks), and to $1 \times 10^{-6}$ and $5 \times 10^{-6}$ for RL training of English and Hindi models, respectively. The loss weights for ML+RL training ($\gamma_1$ and $\gamma_2$) are set to 0.05, and 1.0, respectively. For the dropout p value, 0.5 is used except for the multilingual training (0.3 is used). For hyper-parameters tuning, we try grid-search (e.g., dropout={0.3. 0.5}, learning-rate={$1 \times 10^{-4}$, ..., $1 \times 10^{-6}$}, etc).

| Lang. | Automated Metrics | | | | Task-Specific Metrics | | |
|---|---|---|---|---|---|---|---|
| | B4 | C | M | R | OM | BM | DM |
| Val-Unseen | | | | | | | |
| Eng. | 18.94 | 9.19 | 21.16 | 35.04 | 0.11 | 1.59 | 0.18 |
| Hindi | 23.14 | 8.12 | 29.62 | 35.81 | 0.01 | 1.77 | 0.11 |
| Test-Unseen | | | | | | | |
| Eng. | 17.26 | 6.40 | 21.30 | 34.82 | 0.04 | 1.42 | 0.17 |
| Hindi | 18.98 | 6.69 | 28.47 | 34.53 | 0.03 | 1.52 | 0.11 |

Table A.4: Val-unseen and Test-unseen: the performance of multimodal models on traditional automated metrics and our new task-specific metrics for both English and Hindi dataset (OM: object-match, BM: body-part-match, DM: direction-match).

### A.12.4 Direction-Match Metric

We use the word order heuristic to extract (body-part, direction) pairs to compute direction-match. Our method can match 86% and 87% of human-extracted pairs for English and Hindi, respectively, meaning our metric is very closely matched with how humans would extract (body-part, direction) pairs.

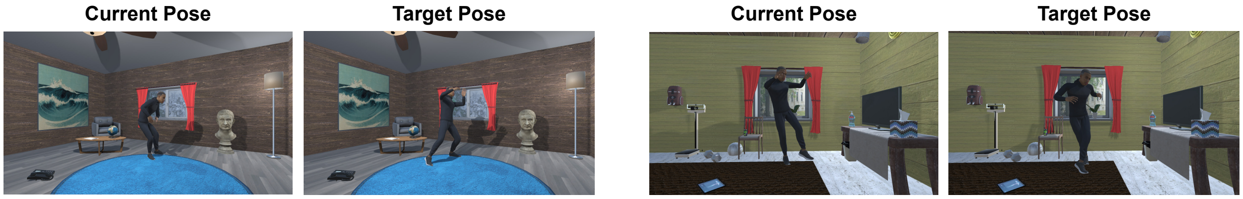
### A.12.5 Unimodal Model Setup

In the pose-correction-captioning task, the V-only model is not fed with the previous token at each decoding time step and does not attend to any previous tokens to decode the next token, and the L-only model does not take as input image pairs. In the target-pose-retrieval task, the V-only model selects the “target” image only by comparing the “current” image to distractors without the correctional description, the L-only model selects the “target” image by comparing the correctional description to distractors without relying on the “current” image.

## A.13 FIXMYPOSE: Results

### A.13.1 Output Examples

Outputs from our V+L multimodal models are presented in Fig. A.18. Our multimodal English model captures the movement of the character’s legs and arms (“bring your right foot to



**Predicted:** you need to bring your right foot to the right and then finally bring your right arm up to be at shoulder height and your right hand up in front of your face

**Predicted Lang:** move your right foot to the right a little towards the sofa turn your body to the left towards the window move your right hand up with palm facing the floor move your left hand up beside your chest

**Predicted Vis:** turn your right foot to your right and from the your body and and the left turn your head towards the the and your head head the window raise your head head head and move your head head the the

**Ground Truth 1:** pull your left foot in right next to your right foot extend your right foot out about 2 feet opposite the direction of the right curtain on the window lift up both hands so that they are in front of your face about a foot from each other and a foot from your face

**Ground Truth 2:** you need to bring your right foot to the right and have that leg slightly straightened you also need to have your back more up right. then finally bring your head to face more forwards then place both your hands up at head height but keep your elbows at the side

**Ground Truth 3:** move your right foot to the right towards the telephone bring your body and head back towards the coffee table and lean to the window move your hands up in front of your head.

**Predicted:** अपने बाएं पैर को अपने दाहिने पैर के सामने ले जाएं अपने दाहिने पैर को थोड़ा सीधा करें अपने ऊपरी शरीर को बाईं ओर थो ड़ा मोड़ें अपने सिर को खिड़की से थोड़ी दूर दाईं ओर ले जाएं अपनी बाहों को नीचे लाएं और अपने हाथों को छाती के स्तर के बारे में ले जाएं।

**Predicted Lang:** अपने दाहिने पैर को अपने दाहिने पैर के पीछे ले जाएं और अपने बाएं पैर को थोड़ा सा सीधा करें। अपने दाहिने हाथ को अपन` शरीर के सामने लाएं और अपने बाएं हाथ को अपने शरीर के सामने रखें।

**Predicted Vis:** अपने दाहिने पैर को पीछे ले जाएं अपने अपने शरीर को सामने अपने अपने शरीर को सामने से लाएं अपने शरीर के के में रखें। अपने शरीर के के के के के के के के के हुए

**Ground Truth 1:** अपने दाहिने पैर को थोड़ा दायें तरफ फेरें। अपने बाएं पैर को अपने दाहिने पैर के सामने रखें। अपने दोनों हाथों को लगभग 1.5 फीट नीचे कर लें। अपनी हथेलियों को जमीन की ओर रखना चाहिए।

**Ground Truth 2:** अपने बाएं पैर को हवा में अपने बाएं पैर के सामने दाईं ओर लाएं अपने कंधे और सिर को थोड़ा नीचे करें अपने हाथों को अ पनी छाती के सामने लाएं आपका ऊपरी शरीर और सिर टेलीविजन की तरफ झुकना चाहिए।

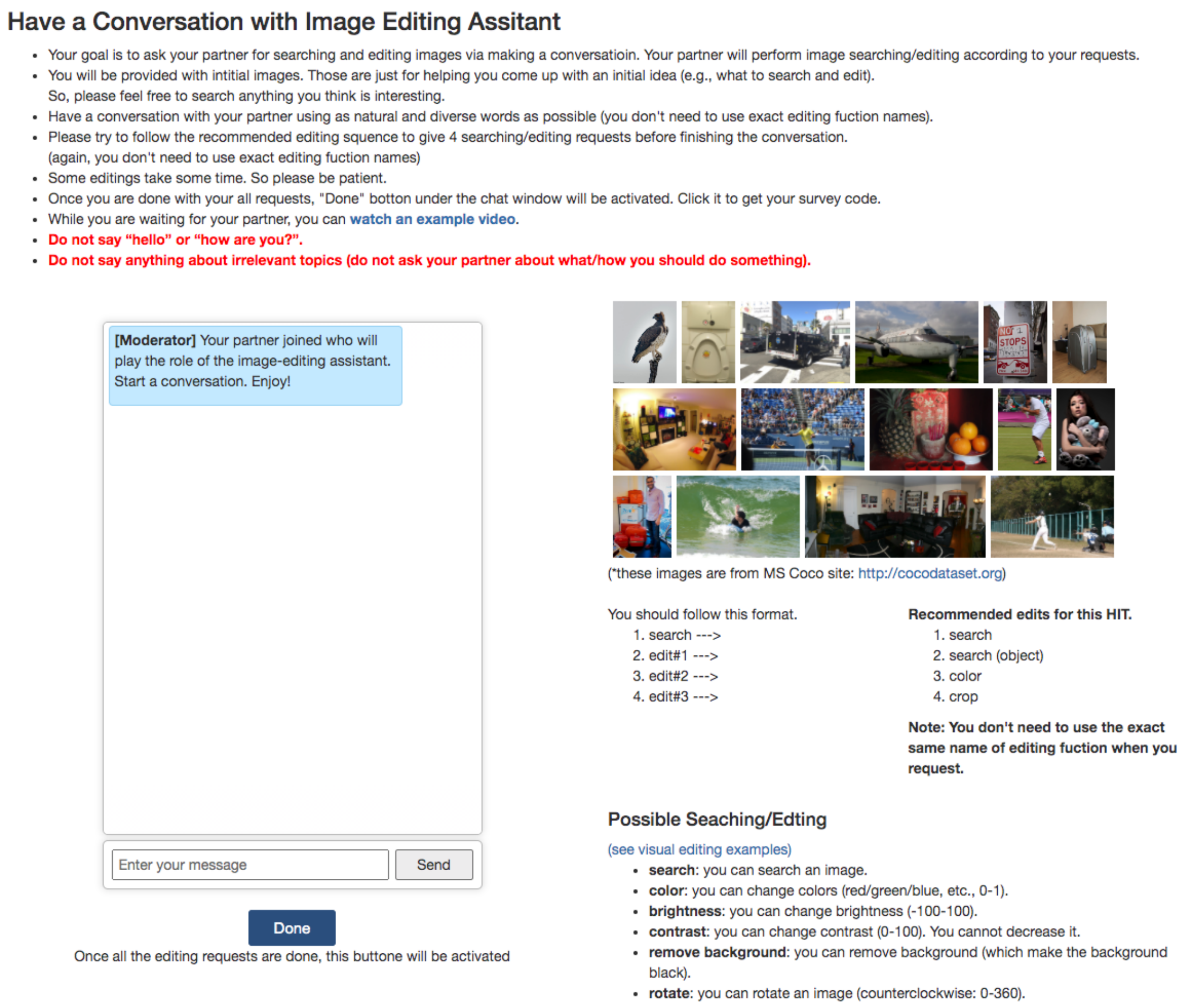
**Ground Truth 3:** अपने बाएं पैर को जमीन पर रखें और इसे अपने दाहिने पैर के ऊपर से पार करें। अपने ऊपरी शरीर को बाईं ओर शीर्षक दें और अ पनी बाहों को तब तक नीचे रखें जब तक वे छाती की ऊँचाई के आसपास न हों।

Figure A.18: Output examples of our unimodal and multimodal models in English (left) and Hindi (right). "Predicted" shows the V+L model output while "Predicted Lang" and "Predicted Vis" show the unimodal outputs for L-only and V-only models, respectively.

the right" and "bring your right arm up to be at shoulder height ... right hand up in front of your face"). The Hindi model captures movement of the body parts and their spatial relationship to each other (English translation: "move your left leg in front of your right leg..."), the model can also describe movement using object referring expressions (English translation: "...move your head slightly away from the window..."). See Fig. A.18 for the original Hindi. For all of the unimodal models, the outputs perform poorly and do not accurately match the image pair. For the V-only models' outputs, the grammar and sentence structure are also very poor.

### A.13.2 "Unseen" Split Results

Table A.4 shows our V+L models' scores on the val-unseen and the test-unseen splits (the scores are chosen by the best performance on the val-unseen split). We suggest that model tun-



Have a Conversation with Image Editing Assitant

- Your goal is to ask your partner for searching and editing images via making a conversatioin. Your partner will perform image searching/editing according to your requests.
- You will be provided with intitial images. Those are just for helping you come up with an initial idea (e.g., what to search and edit).
  So, please feel free to search anything you think is interesting.
- Have a conversation with your partner using as natural and diverse words as possible (you don't need to use exact editing fuction names).
- Please try to follow the recommended editing squence to give 4 searching/editing requests before finishing the conversation.
  (again, you don't need to use exact editing fuction names)
- Some editings take some time. So please be patient.
- Once you are done with your all requests, "Done" botton under the chat window will be activated. Click it to get your survey code.
- While you are waiting for your partner, you can **watch an example video**.
- **Do not say "hello" or "how are you?".**
- **Do not say anything about irrelevant topics (do not ask your partner about what/how you should do something).**

**[Moderator]** Your partner joined who will play the role of the image-editing assistant. Start a conversation. Enjoy!

Once all the editing requests are done, this buttone will be activated

(*these images are from MS Coco site: http://cocodataset.org)

You should follow this format.
1. search --->
2. edit#1 --->
3. edit#2 --->
4. edit#3 --->

**Recommended edits for this HIT.**
1. search
2. search (object)
3. color
4. crop

**Note: You don't need to use the exact same name of editing fuction when you request.**

Possible Seaching/Edting

(see visual editing examples)

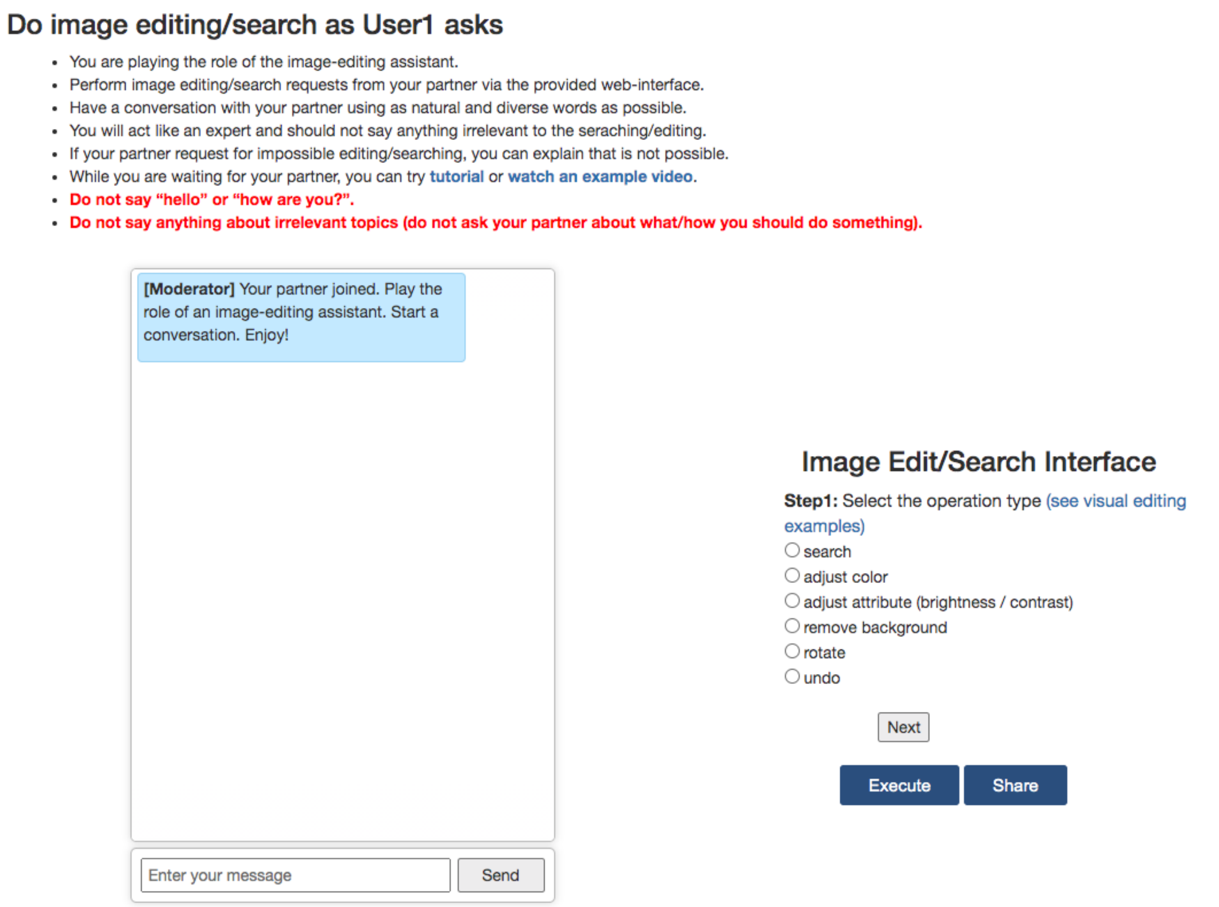
- **search**: you can search an image.
- **color**: you can change colors (red/green/blue, etc., 0-1).
- **brightness**: you can change brightness (-100-100).
- **contrast**: you can change contrast (0-100). You cannot decrease it.
- **remove background**: you can remove background (which make the background black).
- **rotate**: you can rotate an image (counterclockwise: 0-360).

Figure A.19: The data collection interface for user annotators.

ing/selection be done on the val-seen/unseen splits and the results from the test-unseen are reported, following the practice of Anderson et al. (2018b).

## A.14 Caise: Data Collection Interface

Figure A.19 shows the data collection interface of user-annotators. They are provided with initial seed images and a list of suggestions for image search/editing functions. Figure A.20 shows the assistant-annotators' interface. They are provided with tools for image search and editing functions.



Do image editing/search as User1 asks

- You are playing the role of the image-editing assistant.
- Perform image editing/search requests from your partner via the provided web-interface.
- Have a conversation with your partner using as natural and diverse words as possible.
- You will act like an expert and should not say anything irrelevant to the seraching/editing.
- If your partner request for impossible editing/searching, you can explain that is not possible.
- While you are waiting for your partner, you can try **tutorial** or **watch an example video**.
- **Do not say "hello" or "how are you?".**
- **Do not say anything about irrelevant topics (do not ask your partner about what/how you should do something).**

**[Moderator]** Your partner joined. Play the role of an image-editing assistant. Start a conversation. Enjoy!

Image Edit/Search Interface

**Step1:** Select the operation type (see visual editing examples)

○ search
○ adjust color
○ adjust attribute (brightness / contrast)
○ remove background
○ rotate
○ undo

Figure A.20: The data collection interface for assistant annotators.

## A.15 CAISE: Annotation Quality Control

To ensure the annotations maintain high quality, we train and monitor the annotations through an intensive training session. Specifically, one of the authors of this paper is directly paired with some representative annotators (we only know the names of the representative annotators and do not know any private information of them and the other annotators) and collects dozens of dialogues for practice and trains them by correcting every single error. Then the representative annotators train other annotators and collect practice dialogues from each pair of the annotators. We check all the practice dialogues manually and give feedback. We perform this training session multiple times until the quality of the dialogues gets above some threshold (i.e., until the dialogues have no obvious/critical issue).

### A.16 CAISE: Training Details (Reproducibility)

All the experiments are run on a Ubuntu 16.04 system using the NVIDIA GeForce GTX 1080 Ti GPU and Intel Xeon CPU E5-2630. We use PyTorch (Paszke et al., 2017) to build models. We use 512 as the hidden size, 256 as the word embedding dimension, and 2048 as the visual feature (Faster R-CNN) dimension. We use Adam (Kingma and Ba, 2015) as the optimizer with the learning rate $1 \times 10^{-4}$. For dropout p values, 0.3 and 0.5 are used (each for different layers). We run 500 epochs (each epoch takes around 1min 10secs including training+evaluation) for each experiment for the model selection. We use 2020/2021/2022 as random seed values. All the scores are the average values from the run with the three different seeds. The number of trainable parameters of our full model is 11.4M. We use manual tuning (e.g, learning-rate=$\{1 \times 10^{-4}$, ..., $1 \times 10^{-6}\}$, dropout=$\{0.3.\ 0.5\}$, etc.) for choosing hyperparameter values (based on validation scores).

# REFERENCES

Abbasi, B., Monaikul, N., Rysbek, Z., Di Eugenio, B., and Žefran, M. (2019). A multimodal human-robot interaction manager for assistive robots. In *IROS*, pages 6756–6762. IEEE.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. (2018a). Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, pages 6077–6086.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018b). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Andriluka, Pishchulin, Gehler, and Schiele (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*.

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., and Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. In *CVPR*, pages 5167–5176.

Anne Hendricks, L., Wang, O., Shechtman, E., Sivic, J., Darrell, T., and Russell, B. (2017). Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5803–5812.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*, pages 65–72.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. (2016). Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Belz, A. and Reiter, E. (2006). Comparing automatic and human evaluation of nlg systems. In *EACL*.

Berg, M., Bayazit, D., Mathew, R., Rotter-Aboyoun, A., Pavlick, E., and Tellex, S. (2020). Grounding language to landmarks in arbitrary outdoor environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 208–215. IEEE.

Bisk, Y., Marcu, D., and Wong, W. (2016). Towards a dataset for human computer communication via grounded language acquisition. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Bisk, Y., Shih, K. J., Choi, Y., and Marcu, D. (2018). Learning interpretable spatial operations in a rich 3d blocks world. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Blukis, V., Misra, D., Knepper, R. A., and Artzi, Y. (2018). Mapping navigation instructions to continuous control actions with position-visitation prediction. In *Conference on Robot Learning*, pages 505–518.

Blukis, V., Terme, Y., Niklasson, E., Knepper, R. A., and Artzi, Y. (2019). Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on Robot Learning*, pages 1415–1438.

Brahmbhatt, S. and Hays, J. (2017). Deepnav: Learning to navigate large cities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5193–5202.

Brodeur, S., Perez, E., Anand, A., Golemo, F., Celotti, L., Strub, F., Rouat, J., Larochelle, H., and Courville, A. (2017). Home: a household multimodal environment. In *NeurIPS 2017's Visually-Grounded Interaction and Language Workshop*.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., and Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *TPAMI*.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Chen, H., Suhr, A., Misra, D., Snavely, N., and Artzi, Y. (2019). Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Conference on Computer Vision and Pattern Recognition*.

Chen, J., Shen, Y., Gao, J., Liu, J., and Liu, X. (2018). Language-based image editing with recurrent attentive models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8721–8729.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Cirik, V., Zhang, Y., and Baldridge, J. (2018). Following formulaic map instructions in a street simulation environment. In *2018 NeurIPS Workshop on Visually Grounded Interaction and Language*, volume 1.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2018). Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017). Visual Dialog. In *CVPR*.

de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., and Kiela, D. (2018). Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

El-Nouby, A., Sharma, S., Schulz, H., Hjelm, D., Asri, L. E., Kahou, S. E., Bengio, Y., and Taylor, G. W. (2019). Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10304–10312.

Forbes, M., Kaeser-Chen, C., Sharma, P., and Belongie, S. (2019). Neural naturalist: Generating fine-grained image comparisons. In *EMNLP*, Hong Kong.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., Berg-Kirkpatrick, T., Saenko, K., Klein, D., and Darrell, T. (2018). Speaker-follower models for vision-and-language navigation. In *NeurIPS*.

Fu, T.-J., Wang, X. E., Grafton, S., Eckstein, M., and Wang, W. Y. (2020). Sscr: Iterative language-based image editing via self-supervised counterfactual reasoning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. (2016). Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468.

Gao, J., Sun, C., Yang, Z., and Nevatia, R. (2017). Tall: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5267–5275.

Garcia, N., Otani, M., Chu, C., and Nakashima, Y. (2020). Knowit vqa: Answering knowledge-based questions about videos. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Hao, W., Li, C., Li, X., Carin, L., and Gao, J. (2020). Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., et al. (2017). Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*.

Hermann, K. M., Malinowski, M., Mirowski, P., Banki-Horvath, A., Anderson, K., and Hadsell, R. (2020). Learning to follow directions in street view. *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 47:853–899.

Hong, Y., Wu, Q., Qi, Y., Rodriguez-Opazo, C., and Gould, S. (2021). A recurrent vision-and-language bert for navigation. *CVPR*.

Ilharco, G., Jain, V., Ku, A., Ie, E., and Baldridge, J. (2019). Effective and general evaluation for instruction conditioned navigation using dynamic time warping. *NeurIPS Visually Grounded Interaction and Language Workshop*.

Jain, V., Magalhaes, G., Ku, A., Vaswani, A., Ie, E., and Baldridge, J. (2019). Stay on the Path: Instruction Fidelity in Vision-and-Language Navigation. In *Proc. of ACL*.

Jang, Y., Song, Y., Yu, Y., Kim, Y., and Kim, G. (2017). Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2758–2766.

Jhamtani, H. and Berg-Kirkpatrick, T. (2018). Learning to describe differences between pairs of similar images. In *EMNLP*, pages 4024–4034.

Johnson, J., Karpathy, A., and Fei-Fei, L. (2016). Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*. doi:10.5244/C.24.12.

Johnson, S. and Everingham, M. (2011). Learning effective human pose estimation from inaccurate annotation. In *CVPR*, pages 1465–1472. IEEE.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece. IEEE. The best paper award.

Kim, H. and Bansal, M. (2019). Improving visual question answering by referring to generated paragraph captions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Kim, H., Kim, D. S., Yoon, S., Dernoncourt, F., Bui, T., and Bansal, M. (2022). Caise: Conversational agent for image search and editing. In *AAAI*.

Kim, H., Li, J., and Bansal, M. (2021a). Ndh-full: Learning and evaluating navigational agents on full-length dialogue. In *EMNLP*.

Kim, H., Tang, Z., and Bansal, M. (2020a). Dense-caption matching and frame-selection gating for temporal localization in videoqa. In *ACL*.

Kim, H., Zala, A., Burri, G., and Bansal, M. (2021b). Fixmypose: Pose correctional captioning and retrieval. In *AAAI*.

Kim, H., Zala, A., Burri, G., Tan, H., and Bansal, M. (2020b). Arramon: A joint navigation-assembly instruction interpretation task in dynamic environments. In *EMNLP Findings*.

Kim, J., Ma, M., Kim, K., Kim, S., and Yoo, C. D. (2019a). Gaining extra supervision via multi-task learning for multi-modal video question answering. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Kim, J., Ma, M., Kim, K., Kim, S., and Yoo, C. D. (2019b). Progressive attention memory network for movie story question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8337–8346.

Kim, K., Boelling, L., Haesler, S., Bailenson, J., Bruder, G., and Welch, G. F. (2018). Does a digital assistant need a body? the influence of visual embodiment and social behavior on the perception of intelligent virtual agents in ar. In *ISMAR*, pages 105–114. IEEE.

Kim, K., de Melo, C. M., Norouzi, N., Bruder, G., and Welch, G. F. (2020c). Reducing task load with an embodied intelligent virtual assistant for improved performance in collaborative decision making. In *2020 IEEE VR*.

Kim, K.-M., Heo, M.-O., Choi, S.-H., and Zhang, B.-T. (2017). Deepstory: video story qa by deep embedded memory networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2016–2022. AAAI Press.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Krause, J., Johnson, J., Krishna, R., and Fei-Fei, L. (2017). A hierarchical approach for generating descriptive image paragraphs. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 3337–3345. IEEE.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*.

Ku, A., Anderson, P., Patel, R., Ie, E., and Baldridge, J. (2020). Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412.

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Lee, H.-Y., Yang, X., Liu, M.-Y., Wang, T.-C., Lu, Y.-D., Yang, M.-H., and Kautz, J. (2019). Dancing to music. In *NeurIPS*, pages 3586–3596. Curran Associates, Inc.

Lei, Yu, Berg, and Bansal (2020a). Tvr: A large-scale dataset for video-subtitle moment retrieval. In *ECCV*.

Lei, J., Yu, L., Bansal, M., and Berg, T. L. (2018). Tvqa: Localized, compositional video question answering. In *EMNLP*.

Lei, J., Yu, L., Berg, T. L., and Bansal, M. (2020b). Tvqa+: Spatio-temporal grounding for video question answering. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Li, H., Wang, P., Shen, C., and Hengel, A. v. d. (2019). Visual question answering as reading comprehension. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, S., Scalise, R., Admoni, H., Rosenthal, S., and Srinivasa, S. S. (2016). Spatial references and perspective in natural language instructions for collaborative manipulation. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 44–51. IEEE.

Liang, X., Hu, Z., Zhang, H., Gan, C., and Xing, E. P. (2017). Recurrent topic-transition gan for visual paragraph generation. *arXiv preprint arXiv:1703.07022*.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Lin, T.-H., Bui, T., Kim, D. S., and Oh, J. (2018). A multimodal dialogue system for conversational image editing. *Workshop in NeurIPS*.

Lin, T.-H., Rudnicky, A., Bui, T., Kim, D. S., and Oh, J. (2020). Adjusting image attributes of localized regions with low-level dialogue. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 405–412.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*. Springer.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2017). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*, pages 375–383.

Lu, J., Yang, J., Batra, D., and Parikh, D. (2016). Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

MacMahon, M., Stankiewicz, B., and Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4.

Maharaj, T., Ballas, N., Rohrbach, A., Courville, A., and Pal, C. (2017). A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6884–6893.

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., and Batra, D. (2020). Improving vision-and-language navigation with image-text pairs from the web. In *European Conference on Computer Vision*, pages 259–274. Springer.

Manuvinakurike, R., Brixey, J., Bui, T., Chang, W., Artstein, R., and Georgila, K. (2018a). Dialedit: Annotations for spoken conversational image editing. In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 1–9.

Manuvinakurike, R., Brixey, J., Bui, T., Chang, W., Kim, D. S., Artstein, R., and Georgila, K. (2018b). Edit me: A corpus and a framework for understanding natural language image editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Manuvinakurike, R., Bui, T., Chang, W., and Georgila, K. (2018c). Conversational image editing: Incremental intent identification in a new dialogue task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–295.

Mei, H., Bansal, M., and Walter, M. R. (2016). Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Melas-Kyriazi, L., Rush, A., and Han, G. (2018). Training for diversity in image paragraph captioning. *EMNLP*.

Miao, Y. and Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328.

Mirowski, P., Grimes, M., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., Zisserman, A., Hadsell, R., et al. (2018). Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, pages 2419–2430.

Misra, D., Bennett, A., Blukis, V., Niklasson, E., Shatkhin, M., and Artzi, Y. (2018). Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2667–2678.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Mooney, R. J. (2008). Learning to connect language and perception. In *AAAI*, pages 1598–1601.

Nguyen, K. and Daumé III, H. (2019). Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nguyen, K., Dey, D., Brockett, C., and Dolan, B. (2019). Vision-based navigation with language-based assistance via imitation learning with indirect intervention. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Niu, T. and Bansal, M. (2019). Automatically learning data augmentation policies for dialogue tasks. In *EMNLP*.

Novikova, Dušek, Curry, and Rieser (2017). Why we need new evaluation metrics for nlg. In *EMNLP*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Park, D. H., Darrell, T., and Rohrbach, A. (2019). Robust change captioning. In *ICCV*, pages 4624–4633.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *ICLR*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, pages 2641–2649.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. (2018). Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502.

Qi, Y., Wu, Q., Anderson, P., Wang, X., Wang, W. Y., Shen, C., and van den Hengel, A. (2020). Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Reiter, E. (2018). A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401.

Reiter, E. and Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4).

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Rong, Y., Shiratori, T., and Joo, H. (2020). Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration. *arXiv preprint arXiv:2008.08324*.

Saunders, B., Camgoz, N. C., and Bowden, R. (2020). Progressive transformers for end-to-end sign language production. *ECCV 2020*.

Savva, M., Chang, A. X., Dosovitskiy, A., Funkhouser, T., and Koltun, V. (2017). MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. (2019). Habitat: A platform for embodied ai research. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9339–9347.

Scott, D. and Moore, J. (2007). An nlg evaluation competition? eight reasons to be cautious. In *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*, pages 22–23.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Sellam, T., Das, D., and Parikh, A. P. (2020). Bleurt: Learning robust metrics for text generation. In *ACL*.

Seo, M. J., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2017). Bidirectional attention flow for machine comprehension. In *ICLR*.

Serban, Sordoni, Lowe, Charlin, Pineau, Courville, and Bengio (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Shah, P., Fiser, M., Faust, A., Kew, C., and Hakkani-Tur, D. (2018). Follownet: Robot navigation by following natural language directions with deep reinforcement learning. In *Third Machine Learning in Planning and Control of Robot Motion Workshop at ICRA*.

Shi, J., Xu, N., Bui, T., Dernoncourt, F., Wen, Z., and Xu, C. (2020). A benchmark and baseline for language-driven image editing. *Asian Conference on Computer Vision (ACCV)*.

Shinagawa, S., Yoshino, K., Sakti, S., Suzuki, Y., and Nakamura, S. (2017). Interactive image manipulation with natural language instruction commands. *NeurIPS Workshop*.

Shlizerman, E., Dery, L., Schoen, H., and Kemelmacher-Shlizerman, I. (2018). Audio to body dynamics. In *CVPR*.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020). Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33.

Suhr, A., Zhou, S., Zhang, A., Zhang, I., Bai, H., and Artzi, Y. (2019). A corpus for reasoning about natural language grounded in photographs. In *ACL*, pages 6418–6428.

Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *CVPR*, pages 5693–5703.

Tan, H. and Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Tan, H., Dernoncourt, F., Lin, Z., Bui, T., and Bansal, M. (2019a). Expressing visual relationships via language. In *ACL*, pages 1873–1883.

Tan, H., Yu, L., and Bansal, M. (2019b). Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621.

Tang, T., Jia, J., and Mao, H. (2018). Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis. In *ACM Multimedia*.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. (2016). Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640.

Tellex, S. A., Kollar, T. F., Dickerson, S. R., Walter, M. R., Banerjee, A., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. (2019). Vision-and-dialog navigation. In *Conference on Robot Learning (CoRL)*.

Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *CVPR*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*, pages 5998–6008.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Verma, M., Kumawat, S., Nakashima, Y., and Raman, S. (2020). Yoga-82: a new dataset for fine-grained classification of human poses. In *CVPR Workshops*, pages 1038–1039.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Wang, I., Smith, J., and Ruiz, J. (2019a). Exploring virtual agents for augmented reality. In *CHI*, pages 1–12.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2016a). Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer.

Wang, S. I., Liang, P., and Manning, C. D. (2016b). Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378.

Wang, X., Jain, V., Ie, E., Wang, W. Y., Kozareva, Z., and Ravi, S. (2020). Environment-agnostic multitask learning for natural language grounded navigation. *ECCV*.

Wang, X., Wu, J., Chen, J., Li, L., Wang, Y.-F., and Wang, W. Y. (2019b). Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *ICCV*.

Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *CVPR*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Wiseman, S., Shieber, S. M., and Rush, A. M. (2017). Challenges in data-to-document generation. In *EMNLP*.

Wu, Schuster, Chen, Le, Norouzi, Macherey, Krikun, Cao, Gao, Macherey, et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wu, J., Hu, Z., and Mooney, R. (2019). Generating question relevant captions to aid visual question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3585–3594.

Wu, Y., Wu, Y., Gkioxari, G., and Tian, Y. (2018). Building generalizable agents with a realistic and rich 3d environment. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net.

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079.

Xu, Ba, Kiros, Cho, Courville, Salakhudinov, Zemel, and Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057.

Xu, H. and Saenko, K. (2016). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer.

Yan, C., Misra, D., Bennnett, A., Walsman, A., Bisk, Y., and Artzi, Y. (2018a). Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*.

Yan, S., Xiong, Y., and Lin, D. (2018b). Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*.

Yang, L., Tang, K., Yang, J., and Li, L.-J. (2017). Dense captioning with joint inference and visual context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2193–2202.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016a). Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29.

Yang, Z., Yuan, Y., Wu, Y., Cohen, W. W., and Salakhutdinov, R. R. (2016b). Review networks for caption generation. In *NeurIPS*, pages 2361–2369.

Yao, T., Pan, Y., Li, Y., and Mei, T. (2018). Exploring visual relationship for image captioning. In *ECCV*, pages 684–699.

Yu, Z., Yu, J., Fan, J., and Tao, D. (2017). Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 1821–1830.

Zadeh, A., Chan, M., Liang, P. P., Tong, E., and Morency, L.-P. (2019). Social-iq: A question answering benchmark for artificial social intelligence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8807–8817.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. In *ICLR*.

Zhang, Y., Tan, H., and Bansal, M. (2020). Diagnosing the environment bias in vision-and-language navigation. *arXiv preprint arXiv:2005.03086*.

Zhao, L., Peng, X., Tian, Y., Kapadia, M., and Metaxas, D. N. (2019). Semantic graph convolutional networks for 3d human pose regression. In *CVPR*, pages 3425–3435.

Zhu, W., Hu, H., Chen, J., Deng, Z., Jain, V., Ie, E., and Sha, F. (2020a). Babywalk: Going farther in vision-and-language navigation by taking baby steps. In *ACL*.

Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., and Farhadi, A. (2017). Visual semantic planning using deep successor representations. In *Proceedings of the IEEE international conference on computer vision*, pages 483–492.

Zhu, Y., Groth, O., Bernstein, M., and Fei-Fei, L. (2016). Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004.

Zhu, Y., Zhu, F., Zhan, Z., Lin, B., Jiao, J., Chang, X., and Liang, X. (2020b). Vision-dialog navigation by exploring cross-modal memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10730–10739.

Zhuang, W., Wang, Y., Robinson, J., Wang, C., Shao, M., Fu, Y., and Xia, S. (2020). Towards 3d dance motion synthesis and control. *arXiv preprint arXiv:2006.05743*.